NETWORKING AND COMMUNICATIONS

July/August 1991 Volume 9.4

Programmer Journal

.95 Canada $5.95

Spying on NetBIOS

page 45

Diving into the Modem Pool

page 36

Abrash on Edsun CFG/DAC Animation

p

ts for ng

p

olling SCy

p

MAR/APR 92
511611
RON K UNZ
WALL STREET ANALYTICS INC
33-41 85TH ST
JACKSON HEIGHTS NY 11372

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Interoperability.
Platform Transparency.
Distributed Workflow.
Real WYSIWYG Type.
X-Windows.
Extended DOS.
VCPI.
Margaritas by the sea.

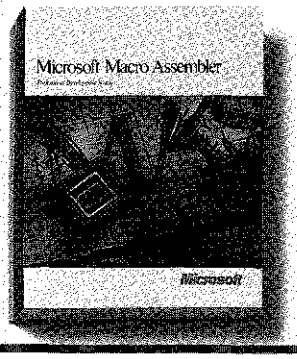
It's just another beautiful day at the Quarterdeck API Conference.

Attend our Fourth Annual API Conference for DESQview and DESQview/X developers and users. August 19-23 in Santa Monica, California. Join major corporations, VARs and developers in learning how to boost user productivity while saving big bucks. Hurry. Call or write today. The conference is filling up fast!

Quarterdeck Office Systems, 150 Pico Blvd., Santa Monica, CA 90405
(213) 392-9851 Fax (213) 399-3802

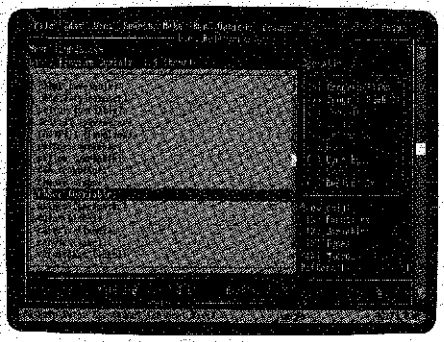
Circle 1 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

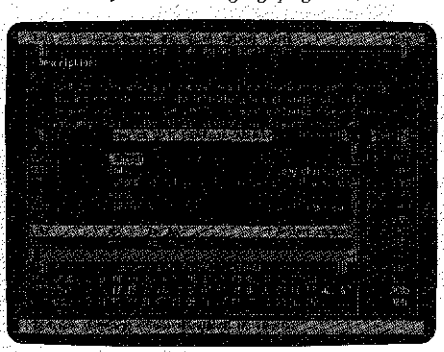


MICROSOFT SOFTWARE DEVELOPMENT TOOLS

Microsoft MASM 6.0. Now 23% less painful.



Source Browser lets you explore the relationships between functions, variables, types, and macros in MASM-only and mixed-language programs.



Microsoft CodeView gives you instant access to all of your source and data, including memory, registers, and flags. You can also access Microsoft Professional Advisor online documentation to get instantaneous assistance.

We know exactly how you feel about assembler: Low-level language, high-level frustration.

But the new Microsoft® MASM 6.0 relieves the stress and strain of Assembler programming (exactly how much depends on how crazed it made you in the first place).

MASM 6.0 has additional high-level language extensions for easier programming. It has the ability to convert C header files into MASM include files, so you don't have to. And the intelligence to optimize the shorter jumps and eliminate NOPs and phase errors from the longer ones.

You also get 386™ /486™ instruction set support.

And a Programmer's WorkBench that supports development of MASM-only or mixed-language programs. All under one seamlessly integrated environment.

So call (800) 541-1261, Dept. Q59, for a free white paper.

And fast relief from headache pain.

For more information inside the 50 United States, call (800) 541-1261, Dept. Q59. Customers in Canada, call (416) 568-3503. Outside the U.S. and Canada, call (206) 936-8661. © 1991 Microsoft Corporation. All rights reserved. Microsoft, the Microsoft logo and CodeView are registered trademarks and Making it all make sense *is a trademark of Microsoft Corporation. 386 and 486 are trademarks of Intel Corporation.*

New Features

- High-level language data types and control structures make mixed language programming easier. Including: PROTO/INVOKE, TYPEDEF, .IF/.THEN/.ELSE, and .REPEAT/.WHILE.
- Macro language extensions such as: @SubStr, @CatStr, @Date, & @Time.
- H2INC converts C header files to Assembler include files.
- Automated Jump shortening speeds execution.
- 486 and expanded 386 instruction set support with flat model (32-bit) support.
- *n*-pass Assembler averts phase errors and Assembler-generated NOPs.
- User-defined Prologue/Epilogue customizeable for Windows.
- /Zm switch plus other features to help accommodate your MASM 5.1 code.
- Significantly improved Assembler speed.
- Programmer's WorkBench v 1.1 and CodeView® v 3.12 make it easier to develop stand-alone .asm and mixed-language programs.
- Source Browser gives instant access to data declarations and function definitions.

PROGRAMMER'S TIPS

The new INVOKE statement automates procedure calling, by handling all pushing and popping of parameters on the stack. Example: INVOKE printf, ADDR fmt1st, ADDR str2, dat1, dat2

If you want MASM 5.1 compatibility, start with the MASM.EXE compatibility driver or use the /Zm switch with ML.EXE.

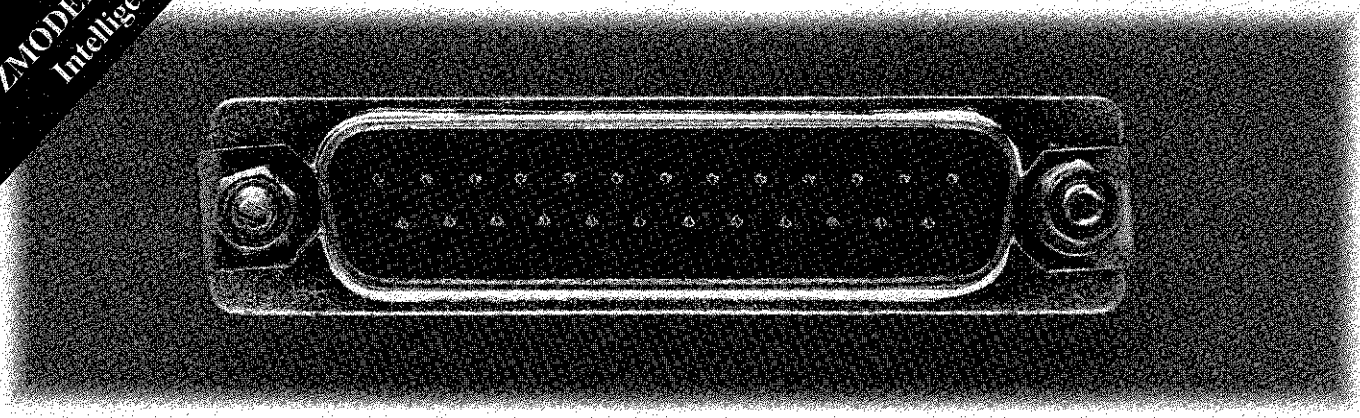
Microsoft
Making it all make sense®

Contact this advertiser directly

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Now Supports: ZMODEM, NS16550 UART, Intelligent Digiboards

Know The Ins and Outs of C Communications



South Mountain Software

Programming communications is a time-consuming, complicated, and sometimes tedious programming job. The task of creating a solid communications application can contain more surprises than a magic show.

Professional C programmers have come to rely on South Mountain for fast, safe and efficient communications functions. With a function library as clean and complete as this one, you don't need to know assembler to have complete RS-232 control.

Functions too Numerous to Mention

We can't list all the many functions here, but our library is designed to support the length and breadth of your application's need. From numerous file transfer protocols to intelligent modem control functions, we do it all. If you don't see what you need in the features box below, call us at **800-451-6174**. We probably have what you're looking for.

Freebies

As an added incentive and learning tool, we include the full source code for a BBS system and a terminal package. Both can be incorporated into projects or dissected for programming hints.

We have also designed and included a sub-system for background file transfers. This procedure can save you hours of experimentation and frustration.

BreakOut II Interactive Communications Debugger

For serious communications application development, BreakOut II will save you hours of debugging time. BreakOut II on a PC will allow you to interact with your comm application on a real-time basis, capturing and transmitting data. In the monitor mode (along with a special cable option) you can monitor the comm session between any two devices on an RS-232 line.

You will be able to spot hardware malfunctions and view line signals up to 115K. All of this for hundreds of dollars less than traditional hardware methods. price $249 includes cable

Real People, Realistic Policies

Our tech support and sales lines are peopled by real C programmers. They are able and willing to answer your questions before and after your purchase.

We offer 30-day, money-back guarantees on all our products, and never charge a royalty or run-time fee. If you've forgotten what it's like to be treated as an important customer, give us a call today. You'll like the change.

Features

- **Interrupt-driven transmit and receive to 115,200 Baud**
- **Simultaneous operation on up to 34 serial ports**
- **Intelligent modem control functions**
- **XMODEM, YMODEM, YMODEM-g, and KERMIT**
- **XON/XOFF and hardware flow control**
- **Concurrent file transfer sessions in foreground/background**
- **Timer and CTRL-Break handling**
- **Extensive multi-port board support**
- **Both Microsoft C and Turbo C support included**

Essential Communications
Price inc. Source $329.
No royalties, 30-Day Money-Back Guarantee

Other Products
C-Utility, Essential Graphics, Essential B-Tree, ScreenStar, /*resident_C*/, Guido, Hold Everything

1-800-451-6174

SOUTH MOUNTAIN SOFTWARE™

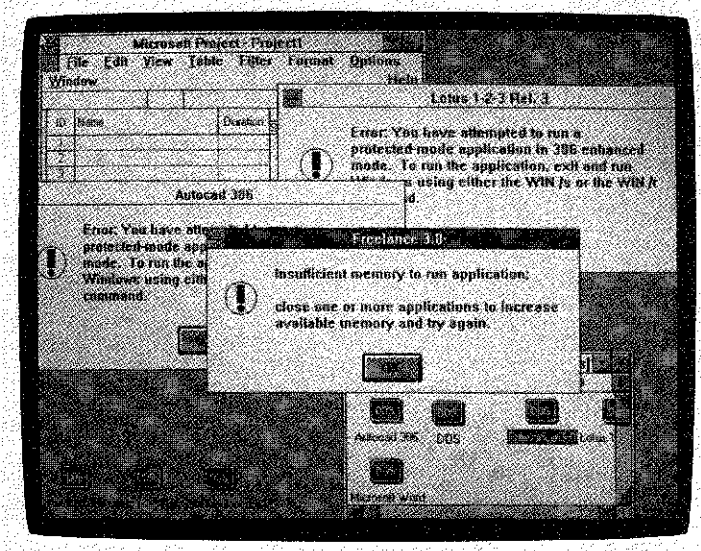
South Mountain Software Inc.
76 S. Orange Ave., S. Orange, NJ 07079
NJ 201-762-6965 Fax 201-762-0118

Essential Graphics, Essential Communications, Essential B-tree C-Utility Library, Hold Everything, Breakout 2 GUIDO, /* resident_C*/ Screen Star are trademarks of South Mountain Software Inc.

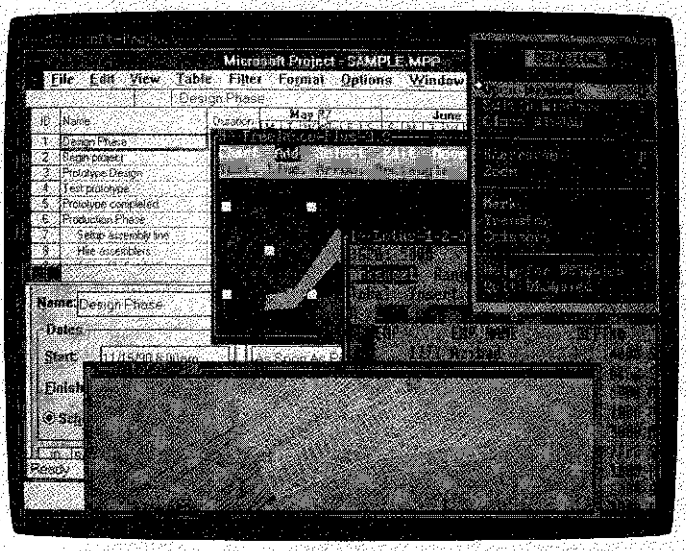
PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Is Windows 3.0 the end of DESQview™?

Not if you're still using DOS programs.


Microsoft Project - Project1
File Edit View Table Filter Format Options Window Help
Lotus 1-2-3 Rel. 3
Error: You have attempted to run a protected-mode application in 386 enhanced mode. To run the application, exit and run
Autocad 386
Freelance 3.0
Insufficient memory to run application.
Close one or more applications to increase available memory and try again.
OK
Autocad 386
DOS
Lotus
Microsoft Word

Windows 3.0. The multitasking, windowing environment.



DESQview 386 2.3. The multitasking, windowing environment.

Are you still using MS-DOS programs on your PC?

You may want to use DESQview as your primary operating environment.

The new DESQview 2.3 and DESQview 386 2.3 let you use your favorite DOS and DOS-extended programs in windows side-by-side on 80286, 80386 and i486 PCs. As you can see above, you can even run Windows programs within DESQview 386. So the next time you get error messages like the ones at the left in Windows, remember how the same set of programs look running in DESQview.

Whatever standard you use—DOS, extended DOS or Windows—DESQview is still the best way to get the most out of the hardware and software you own today.

DESQview. The obvious choice.

Quarterdeck

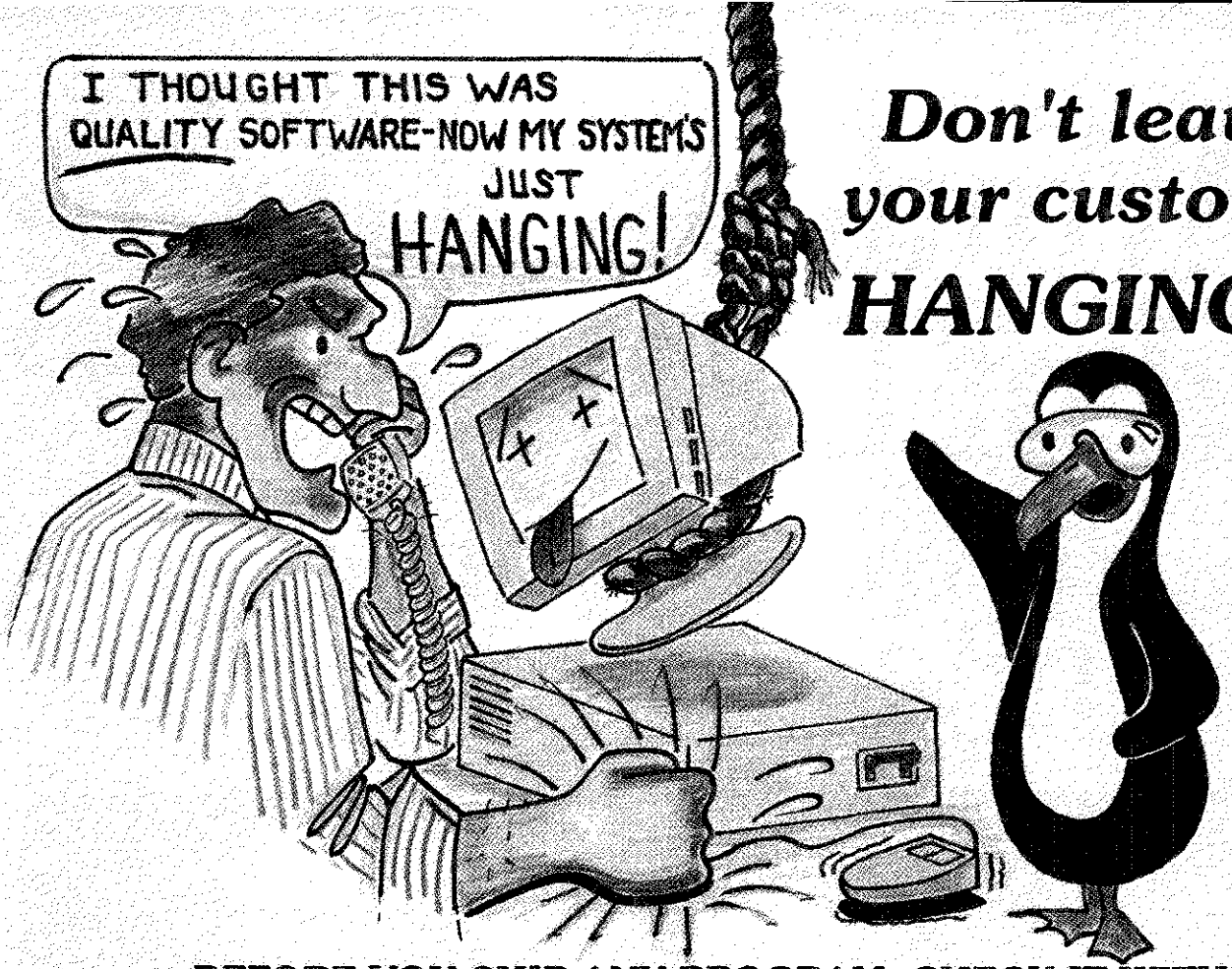
Quarterdeck Office Systems, 150 Pico Boulevard, Santa Monica, CA 90405 (213) 392-9851 Fax: (213) 399-3802

This comparison was made using a system like the one you might run: Both shots show an ALR FlexCache 33/386 running DOS 3.3 with VGA display adaptor, Novell NetWare v3.01 Rev. A, with IPX/SPX v3.01 Rev A, Microsoft Mouse 7.00, and Microsoft SMARTDrive v3.03 disk cache. Buffers were set to 20. For the Windows screen, we ran Microsoft Windows 3 HIMEM.SYS and EMM386.SYS. For the DESQview screen, we ran QEMM 386 v5.1.

Trademarks: Microsoft, Windows, MS-DOS, 80386, i486, ALR, FlexCache, Novell, NetWare. ©1990 Quarterdeck Office Systems

Circle 4 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



Don't leave your customers HANGING. . .

BEFORE YOU SHIP ANY PROGRAM, CHECK IT WITH

COMPUTER LANGUAGE PRODUCT EXCELLENCE AWARD 1990 COLA

BOUNDS-CHECKER™

WINNER PC MAGAZINE BEST OF 1990

JANUARY 15, 1991 PAGE 168

FINDS OUT-OF-BOUNDS MEMORY ACCESSES *AUTOMATICALLY*

Your program may have 10,000 to a million lines of code. It may occasionally hang mysteriously or it may appear to run flawlessly every time. But under DOS, how can you ever be sure that your program is not corrupting memory it does not own? The only way to be 100% sure is to BOUNDS-CHECK before you ship.

To use BOUNDS-CHECKER you build your program with debugging information (we support most compilers including Microsoft, Borland & JPI). Then you just type <BC file-name>. BOUNDS-CHECKER sets up the 386™/i486 for protection and lets your program fly. If your program accesses memory it does not own or overwrites its own code, BOUNDS-CHECKER pops up displaying the offending SOURCE-LINE or instruction.

Programming under DOS is a gamble, so why not stack the odds in your favor--CALL TODAY.

(603) 888-2386

OTHER FINE NU-MEGA PRODUCTS. . .

The ultimate systems debugger.

Debug:
- Interrupt routines
- Device drivers
- T&SRs
- ROMS
- Applications
- Overlays

Soft-ICE™

Features:
- Break out of a hung program
- Real time Break-Points
- Back-Trace history
- Works with other debuggers

If you are debugging an application, Soft-ICE is seamlessly integrated with BOUNDS-CHECKER so you can easily go back and forth between BOUNDS-CHECKing and debugging: a combination many programming professionals can't live without.

1988 BYTE AWARD OF DISTINCTION

Run CODEVIEW for Windows on a single monitor
CV/1.....$129
• Runs in a window • No annoying flash •
• Runs on any display that supports windows •

Call by 4:00 PM EST TODAY and ask us to EXPRESS you an info packet. In most cases you will receive it by 10:30 AM tomorrow. (USA only)

All Nu-Mega products require a 386, 386SX or 486. MS-DOS and Codeview are trademarks of Microsoft Corp. 386 is a registered trademark of Intel Corp. Nu-Mega, BOUNDS-CHECKER, Soft-ICE and CV/1 are trademarks of Nu-Mega Technologies, Inc.

™ Nu-Mega TECHNOLOGIES INC

P.O. Box 7780 • Nashua, NH • 03060-7780 U.S.A.
(603) 888-2386 • Fax (603) 888-2465

BOUNDS-CHECKER $249
Soft-ICE $386
Special Offer
Buy BC & S-ICE .....Save $100

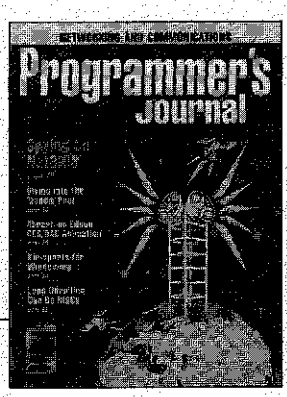
30 Day Money Back Guarantee

Circle 5 on Reader Service Card.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Programmer's Journal™

July/August 1991 Volume 9.4



Hook up to a communications network, where the sky's no limit. Threads of wire and waves of wizardry connect the globe to our unfathomed future. For a look at what lies just beyond the horizon, join us in scouting out the possibilities explored in this issue. So plug in, power on, and find out what it's all about.

Cover Artist: Rob Schuster

FEATURES

GRAPHICS **24** **A COAT OF MANY COLORS** *By Michael Abrash*

Pick a pack of purple pixels . . . or make them red, white, and blue. Graphics guru Michael Abrash shows how to boost your screen's color quotient and program some animation using Edsun's plug-in DAC replacement, the CEG/DAC.

NETWORKING **36** **COMMUNICATING OVER NETBIOS** *By Brett Glass*

There's probably a modem pool in your future. Brett gets you into the swim of things by mixing communications with NetBIOS sessions and Turbo Pascal objects. He also looks briefly at not-so-standard network modem interfaces. So step into the LAN of plenty!

NETWORKING **45** **SPYING ON NETBIOS** *By John Otken*

Need to network your DOS machine with other computers? John develops assembly language routines to interface with NetBIOS and a utility for spying on datagrams. You'll also find his simple but elegant client and server applications for transferring files.

C **55** **WINDOW VIEWPORTS** *By Allen Holub*

Allen continues to perfect the art of window management in this second part of his text-windowing system. He tacks a display layer (or "viewport") on to the previous text buffer routines, creating a stand-alone windowing system that won't tie you or your computer's memory up in knots.

OPTIMIZATION **66** **LOOP UNROLLING: A RISCY BUSINESS** *By Mark J. Barrenechea*

Compiler optimizations tend to remain in the realm of the mysterious. Mark unravels loop unrolling to reveal its uses and discuss its limits. He also compares scalar and vector optimizations on the i860 with loop unrolling on the 80486.

ADA/C **73** **DATA STRUCTURE IMPERATIVES** *By Richard Riehle*

It's embarrassing when you don't know an algorithm from a biorhythm, or which data structure you need to get into the swing of things while traversing a binary tree. Richard poses challenging questions and provides some sample solutions.

BASIC **78** **TEXT FILTERS IN BASIC** *By Murray L. Lesser*

Who says simple can't be beautiful? Once you get to know Murray's text filter program, you'll want to write others—and you'll know how to go about it.

COLUMNS

REDIRECTIONS **18** **PINK NOISE** *By Frank D. Greco*

CONSULTING **22** **DON'T BE WRONG ABOUT YOUR RIGHTS** *By William H. Roetzheim*

ARMCHAIR **84** **NETWORK MADNESS** *By M. Steven Baker*

IMPRESSIONS **86** **NETWORK DEBUGGING AND COMM LIBRARIES** *By M. Steven Baker*

OFFRAMP **96** **BENCHMARK SPECULATIONS** *By Hal W. Hardenbergh*

DEPARTMENTS

6 **EDITOR'S NOTE**

8 **LETTERS**

80 **ADVERTISER INDEX**

91 **PRESSROOM**

93 **PJ BILLBOARD**

95 **BBS LISTINGS**

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

EDITOR'S NOTE

By the time you read this, I should be back swimming in the river. But for now it's cold and rainy here—an ideal time for working indoors with a collection of connected boxes. Although it may never be the "Year of the LAN," networks have become pervasive in the PC realm. The norm in large organizations, networks are filtering down to small businesses as surely as summer follows spring.

ALPHABET SOUP CAUSES NETWORK INDIGESTION

Problems remain with PC connectivity. In DOS, the battle continues between network drivers and applications for memory space, because network software isn't integrated with the operating system. So DOS memory managers are increasing in importance while OS/2 is on the back burner. Contrast this mayhem with the Unix world, where networking is a way of life.

Originally, most PC networks were homogeneous (everything was DOS); now, users are demanding interconnectivity with Macs, Unix boxes, and other machines. They're insisting on seamless, transparent connections, even though some PC protocols, such as NetBIOS, were not designed for internetworking. Also frustrating is the profusion of file-naming conventions and text-file formats. It's unfortunate that DOS designers didn't use Unix's "/" as a directory separator or support case in DOS file names. Macintosh system designers were equally shortsighted—they chose a different separator for directories, and a carriage return to indicate a text end-of-line.

Network protocols offer a large gulp of alphabet soup (aka acronyms). The International Standards Organization (ISO), academics, and the U.S. and European governments have endorsed the ISO "seven-layer cake" model. The PC and Unix marketplace favors well-tested non-ISO protocols, such as TCP/IP and SPX/IPX, that have fewer layers, take up less overhead, and perform better.

DOS EXTENDERS GO DPMI FOR TRANSPARENCY

Microsoft has quietly slipped a DOS extender into their C 6.0*x* package. They also have released Visual Basic and a Fortran update for Windows. Phar Lap and Ergo have introduced new Windows-compatible 286 DOS extenders. Phar Lap's extender supports OS/2-style function calls; other companies also have this OS/2-under-DOS technology. Rational, for example, has released Oxygen, a DOS extender for Microsoft C 6.0*x* tools that binds itself to the OS/2 executables for Microsoft C.

Ergo's DPMI 286 DOS extender allows you to build an executable that runs under Windows 3 as a DPMI client or under Ergo as a DPMI server. Other than Windows 3, Ergo's is the only DPMI server available. Interestingly, I was able to run the Phar Lap 286 DOS extender under Ergo.

Phar Lap has a new version of its 386 DOS extender that is not Windows—read DPMI—compatible and is testing another one that is. Ergo is finishing up a DPMI-compatible, 386 DOS-extender upgrade. Meanwhile, Intel has released its C Code Builder Kit. It's bundled with a DOS extender by IGC that is DPMI-compatible but won't work with EMS managers (not VCPI-compatible). A bit crazy, but soon the DOS-extender world will all be Windows-compatible.

BIG FELLAS HAVE BIG PROBLEMS

Large companies often dislike going "outside" for products or services. Take Microsoft's Excel-for-Windows group, which spends 11 hours doing a build with their Windows linker that takes about 10 minutes with SLR's Optlink/Windows.

As you've no doubt heard, Borland is working on a database competitor to Ashton-Tate's dBASE IV. Is Ashton-Tate—still reeling from the loss of dBASE language ownership—prepared for a direct-marketing onslaught from Borland? If you're wondering from whom Borland might purchase this technology, note that both they and Fox Software are indebted to the same venture capital firm. And of course, these new products will need to support interconnectivity. So will it be Borland-SQL or TurboBASE/LAN?

Steven Baker

Programmer's Journal

Editor
M. Steven Baker
Managing Editor
Cecelia Hagen
Associate Editor
M. Elizabeth Hardinger
Assistant Editor
Deborah Healey
Contributing Editors
Michael Abrash, Brett Glass, Frank D. Greco, John Otken, Tom Swan
Founder
Greg Estes

Publisher
Elizabeth Oakley
Advertising Manager
Jay Moore
Advertising Coordinator
Bonnie Chin
Circulation Manager
Randy Randolph

Production & Art Director
Wanda Walker
Production Assistants
Margie Hanna, Nicole Harris

Financial Controller: Thomas Graham
Subscriptions: Pamela Coleman
Office Staff: Tina Bates, Barbara Curtright, Jill Edwards, Bruce Searl

Business and Editorial Offices: ***Programmer's Journal,*** P.O. Box 70167, Eugene, OR 97401-0110, (503) 747-0800, (800) 234-0386, Fax: (503) 746-0071, BBS: See page 95

Subscriptions: 1 year (6 issues) $29.70

Change of Address: ***Programmer's Journal,*** P.O. Box 70387, Eugene, OR 97401-0121

Articles and letters in *Programmer's Journal* represent the views of the authors and not necessarily those of the publisher. Letters to the editor become the property of *Programmer's Journal* and may be edited for publication.

Note that all code and listings in *Programmer's Journal* appear "as is," and *Programmer's Journal* makes no guarantee or warranty of their quality, performance, accuracy, or suitability for any particular purpose. *Programmer's Journal* assumes no responsibility or liability to any person or entity with respect to any damage caused or alleged to be caused by their use.

Advertising: For ad rates, deadlines, and placement, call Jay Moore at 800-234-0386, or write to *Programmer's Journal*, P. O. Box 70167, Eugene,OR 97401-0110. *Programmer's Journal* is produced using the following software products: Ventura Publisher,® Aldus PC PageMaker,® and Corel Draw. The PostScript™ files are output on a Linotronic™ 200.

Programmer's Journal (ISSN 0747-5861) is published bimonthly by Oakley Publishing Company, 150 N. 4th Street, Springfield, OR 97477-5454. Second-class postage paid at Springfield, OR, and additional mailing offices. USPS # 000 293. **POSTMASTER:** Send address changes to *Programmer's Journal*, P.O. Box 70387, Eugene, OR 97401-0121.

© 1991 *OAKLEY PUBLISHING COMPANY*

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

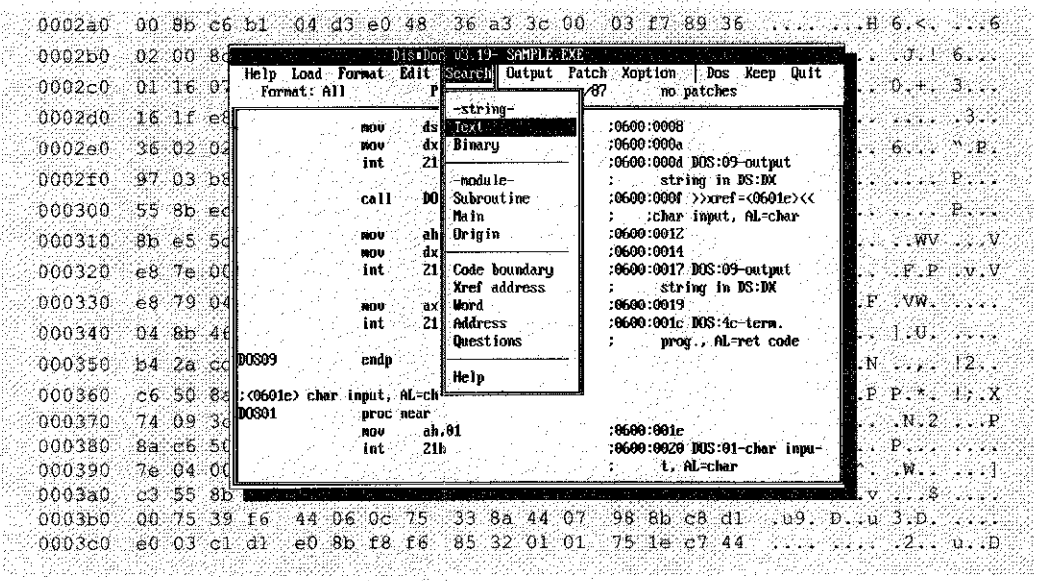
NEW VERSION

No Source?
No Problem.

Automatic BIOS, PSP, MS-DOS interrupt labeling built-in

Batch and Interactive modes

Interactively locates code/data boundaries



8086 to 80486 instruction sets supported

Interactive cross-referencing

Creates listing only after disassembly is perfect

Dis • Doc Professional™

New **Dis • Doc** *Professional* is your **dual-mode** key to any DOS source code. It works in batch and interactive modes simultaneously, allowing you to generate the core information of even the most complex programs **fast**...and modify them even faster. Most programs will come apart in just two minutes. Imagine what you can do with a tool this powerful! **Dis • Doc** sifts through programs **eight times** for guaranteed accuracy. When code gets mixed up with data, our toolbox comes to the rescue with **smart** search and **easy** edit utilities.

Warning: Dis • Doc *Professional* may change the way you work **forever**.

Programmers who used to shy away from fixing outmoded programs with no source code are going to discover a **valuable** new talent: the ability to modify and revise codes that would cost way too much to start over (it's a programming manager's dream). Save your employer huge new-programming fees and enhance your marketability. **Dis • Doc** is so easy to learn, you'll be a high-dollar hero in no time!

Dis • Doc *Professional* is an amazing new teaching tool. Learn how programs work...take them apart and see the writing techniques that top pros use. Use it to assist in debugging. Hunt down viruses and write killers. **Dis • Doc** can save you **years** of frustration, and it only costs $249.95.

COMPARE FOR YOURSELF!

	Dis•Doc™	Sourcer™
Speed of disassembly (bytes/sec)	***822***	*120*
Accuracy (code/data separation*)	***99.9%***	*99.9%*
Unlimited file size	***YES***	*NO*
BIOS labeling	***YES***	*XTRA $$*
BIOS/PSP file labeling	***YES***	*NO*
Unpacks EXE files	***YES***	*XTRA $$*
Patcher	***YES***	*NO*
Batch mode	***YES***	*YES*
Interactive mode	***YES***	*NO*
Time it takes to customize a label	***5 sec.***	*2-5 min.*
Time it takes to change data to code	***5 sec.***	*2-5 min.*

*at eight passes based on a 9040 byte EXE file picked at random

To order your **Dis • Doc Professional™** disassembler toolkit or our $3 demo disk, simply call:

1-800-336-1961 in US
1-203-953-0236 outside US
or send check or money order for **$249.95** plus $5 s&h to:

RJSwantek, Inc.
178 Brookside Road
Newington, CT 06111-1320 USA

For more information call 1-203-953-0236 • FAX available • MasterCard & Visa • Shipped Immediately Via UPS Blue inside US.

Sourcer is a registered trademark of V Communications, Inc.

Circle 6 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

LETTERS

PRODUCTIVITY OR BUST

I really enjoy your magazine and find it invaluable in my work. Thank you for a job well done.

I read William Roetzheim's "Programmer productivity" (*PJ* 9.3) with great interest. I brought the magazine to work, and the first reaction of a fellow programmer was, "Where does this guy live? California?" Sure enough, you listed the author's San Diego address.

I cannot imagine anyone here in western Pennsylvania calling a programmer who produces 2,000 lines of code per month a "guru" or "indispensable," let alone a "super programmer." And someone producing a mere 500 lines of code per month would hardly be considered "professional" here—more likely "unemployed." Yesterday I produced about 2,000 lines of code. I expect to get the same count today.

G. Adam Stanislav
East Pittsburgh, PA

After reading William H. Roetzheim's "Programmer productivity" (*PJ* 9.3), I became somewhat ill of mind and spirit. I find the author's "hammer them hard and

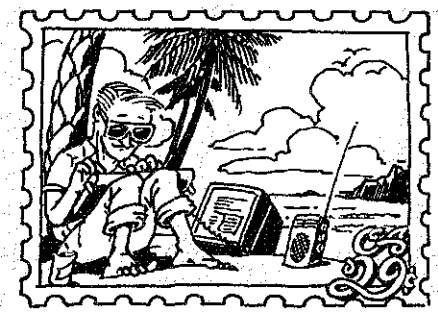


ILLUSTRATION BY JIM CARPENTER

fast" approach to confronting a programmer's honest repetitive errant behavior to be somewhat repugnant. I don't find the policy of managing people by fear and intimidation all that desirable, even if Roetzheim considers it to be more cost-effective in grooming a class of "Super Programmers." (I find that term as disturbing as his suggested methods.) Where's Marley's ghost when you need him?

Phil Pistone
Park Ridge, IL

BACK TO BASIC STRINGS

I read with interest Murray Lesser's article about string handling in BASIC PDS 7 (*PJ* 9.2), but he missed one useful feature of the **StringAddress** function. Not only does it return the address of the far string in registers DX:AX, but it also returns the length of the string in register CX. This makes calling **StringLength** unnecessary. When you consider that the new far string operations are much slower than the older near string operations, you will appreciate the reduced code and added speed in your ASM functions.

Daniel P. Stasinski
Willits, CA

A LODE OF WINDOWS

Paul Yao's interesting article on Windows compilers (*PJ* 9.2) was much appreciated. Inadvertently, however, he may have created the impression that these compilers are otherwise compatible, which isn't so. Borland's C++ 2.0 happily compiled the Windows parts of a little Microsoft C demonstration program I had, but it balked at Microsoft's **_dos_findfirst** function; Borland also uses **findfirst,** and the related data structure is different. The other compilers undoubtedly have similar variations for other-than-standard (i.e.,

AN OPEN LETTER TO SOFTWARE DEVELOPERS

Excessive litigation has been known to put doctors out of business. Now it's the programmer's turn. If you program professionally and develop commercial products that have any level of complexity, prepare to be sued. It is virtually impossible to write a software product without violating dozens of patents, and things are only going to get worse.

The League for Programming Freedom (LPF), a not-for-profit organization of programmers and users, is opposed to software patents. The LPF's position paper covers the wide range of issues regarding software patents (see address below). This paper documents numerous examples of ludicrous patents and of lawsuits—which have been upheld—that affect virtually every large software system.

Software algorithms should not be patented, any more than, say, chess moves. A clever chess move is not obvious, but any bright person faced with the board and some time could come up with it. The same principle is true for software algorithms—yet more than 2,600 software patents have been granted, 599 in 1990 alone.

You may come up with a dozen clever programming solutions in a day—every one of which may quite possibly already be patented. If this flawed system is not remedied, none of us will be able to do our jobs.

As a programmer who has been affected by the new rage of legal terrorism, I want to tell the personal side of the story. It revolves around the work I did to produce sound effects from within a computer game. There is only one way to make sound on an Apple II or an IBM: turn the speaker on, apply 5 volts, and turn the speaker off. The only thing you can vary is the rate at which the speaker is switched on and off. If you toggle the speaker very rapidly, it makes a quieter sound.

I wrote a trivial routine to do this, less than a couple dozen instructions total, and put it into a computer game. I have received two phone calls from two different companies putting me on notice that I may be violating their patents on this complex, high-technology technique covering "pulse-width modulation." So far I haven't been sued, but I've been asked to remove my product because it *might* violate their patent.

What's next? Threatening phone calls because I draw a cursor on the screen? (Patent #4,197,590 is being pursued vigorously by Cadtrak.) Or because I display and scroll some text in a couple of windows? (Patent #4,736,308—ask Apple, which was sued.) What if I optimize a routine that recalculates a table of numbers? (Ask Lotus, which is being sued by Refac Technology Development Corporation, a "technology" company with plenty of lawyers but no programmers or engineers.) Where will it stop?

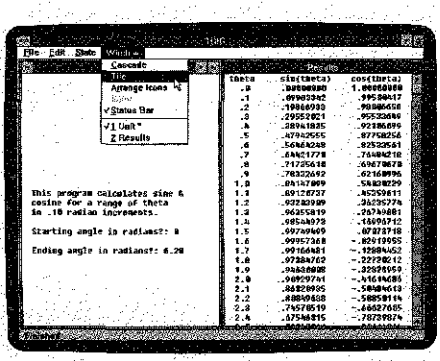
Contact LPF at (617) 243-4091, or send Internet mail to league@prep.ai.mit.edu, or write to them at 1 Kendall Square #143, P.O. Box 9171, Cambridge, MA 02139.

—Name withheld by request

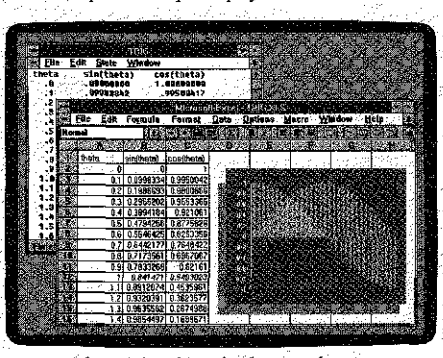
PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

MICROSOFT SOFTWARE DEVELOPMENT TOOLS

Windows helps FORTRAN users create bigger apps.



The Microsoft Windows graphical environment lets you have multiple windows in your FORTRAN applications. This means that you can resize and scroll input and output displays to view data...



...and to analyze data, simply cut and paste your FORTRAN output into programs such as Microsoft Excel for Windows and you can view it instantly in graphical form.

With the new Microsoft® FORTRAN Professional Development System version 5.1, your existing code taps into the power of the Microsoft Windows™ graphical environment. FORTRAN 5.1's new QuickWin library lets you develop 16-bit apps that access greater memory than ever before – breaking the 640K barrier on 286 and 386 machines.

You can use the FORTRAN QuickWin library to take advantage of multiple I/O windows, multitasking, and cut and paste. Or write FORTRAN routines called from mixed-language Windows apps. And FORTRAN 5.1 has Programmer's WorkBench, with a new CodeView® debugger and Source Browser.

To see the advantages of bigger, multi-windowed 16-bit applications, give us a call at (800) 541-1261, Department R38 and order your FORTRAN update today.

**As used herein, "DOS" refers to MS-DOS or PC-DOS operating systems. © 1991 Microsoft Corporation. All rights reserved. Printed in the U.S.A. Inside the 50 United States, call (800) 541-1261, Dept. R38; outside the 50 United States, call (206) 936-8661. Customers in Canada, call (416) 568-3503. Microsoft, MS-DOS, OS/2, CodeView and the Microsoft logo are registered trademarks and Windows and* Making it all make sense *are trademarks of Microsoft Corporation.*

Key Features

- Create Windows .DLLs in FORTRAN using new or existing code.
- QuickWin Features:
  - QuickWin child windows are easily created using the OPEN statement.
  - User-defined positioning and titles for child windows.
  - Automatically generated scroll bars for output that extends past a single screen.
- CodeView debugger supports DOS*, Windows-based and OS/2® applications.
- Extended CodeView debugger for large DOS programs.
- Complete online documentation for the FORTRAN language and all compile and link switches.
- DOS and OS/2 run-time libraries are compatible with other Microsoft languages.
- Floating-point support includes co-processor, emulation, and alternate math libraries.
- 100% ANSI 77 compatibility and numerous IBM®, VAX® and ANSI 8X extensions.
- New BYTE keyword emulates VAX data types.
- Language Extensions include DOUBLE COMPLEX variables, precision and maxexponent inquiries.

PROGRAMMER'S TIPS

- *Use the new /MW option with the FL command to invoke the QuickWin library. For example: "FL/MW MYAPP.FOR" is all it takes to make MYAPP a Windows-based program.*
- *Use the ALLOCATE statement to dynamically size arrays and to access more than 16MB of memory on a 386.™*

Microsoft
Making it all make sense®

Contact this advertiser directly

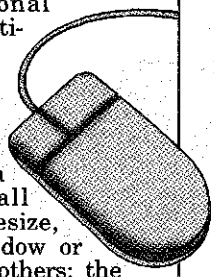
PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

C TOOLS PLUS/6.0

Integrate sophisticated features into your Microsoft C and QuickC applications.

FROM BLAISE COMPUTING

C TOOLS PLUS version 6.0 is filled with many advanced routines for developing high-powered C applications, including: virtual, stackable menus and windows with full mouse support and optional "drop shadows"; multiple virtual pop-up help screens; a miniature multi-line editor for gathering user responses in a robust fashion; a single function call which can move, resize, and promote a window or menu on top of all others; the ability to update covered windows automatically when they are written to; support for EGA, VGA, and MCGA text modes including 30-, 43-, and 50-line modes; support for the enhanced (101/102 key) keyboard.



All this and more for only $149!

C TOOLS PLUS/6.0 also contains functions for writing interrupt service routines; creating pop-up memory resident applications; general memory "peeks" and "pokes"; access to the DOS PRINT utility; as well as many other general utility functions and macros.

COMPLETE PROFESSIONAL PACKAGE.

Blaise Computing's function libraries offer easy to use solutions to your programming needs. You get source code, complete sample programs, and a comprehensive reference manual with extensive examples. Supports QuickC and Microsoft C 5.0 and later.

30 DAY GUARANTEE.

If during the first 30 days you are not completely satisfied, we'll refund your money.

Other powerful products from Blaise Computing Inc.

Win++	$249
Turbo Vision Dev. Toolkit (Pascal)	$149
Turbo C TOOLS	$149
C ASYNCH MANAGER	$189
ASYNCH PLUS	$189
View232	$189

Call today for more information

(800) 333-8087

BLAISE COMPUTING INC.

819 Bancroft Way
Berkeley, CA 94710
(415) 540-5441

FAX (415) 540-1938

Trademarks are property of their respective holders.

Circle 8 on Reader Service Card

not Unix-like) functions: the existing level of incompatibility among these products is not magically improved by Windows.

This, of course, only adds to the fun. As I've begun my first adventures in Windows, I'm developing a warm appreciation for Bill Gates's splendid "Programmers' Full Employment Project." I was planning alternate careers for my Golden Years, and just in time there comes this potentially lucrative *and* technically challenging gold mine!

J. G. Owen
Fort Salonga, NY

GUI, YES—GRAPHICS, NO

The March/April issue (*PJ* 9.2) was nearly perfect. Having once been a happy subscriber to *Micro Cornucopia*, I was delighted to see Dave Thompson in your pages. The articles by Thomas Lahey and Kim Kokkonen were particularly useful.

In "Porting to the Mac," I believe W. Sumerlin left out several useful books (and maybe a compiler). I once had to write a front end to FoxBase/Mac and wound up doing it in Turbo Pascal for the Mac. This is a terrific compiler, regrettably not much supported by Borland. Now that Apple has introduced machines resembling cheap Macs, maybe Borland will think again. My students are using Think Pascal 3.0, but for me Turbo is much less hassle, despite the lack of good debugging support. The most helpful books for learning Mac programming were Borland's own documentation for Turbo and the Turbo Tutor for the Mac. A close runner-up was Chernicoff's *Macintosh Revealed* series (volumes 1 and 2), essentially a humane rewrite of *Inside Macintosh*.

If Turbo Vision is a glimpse of things to come for Turbo Pascal for Windows, then Borland's C++ is probably the better choice. The idea of Turbo Vision is great, but it was obviously implemented by someone who was in love with LISP. I recommend that anyone pursuing text-based GUI invest the $100 or so in Kokkonen's Object Professional, and forget Turbo Vision. To be honest, I don't plan to run Windows before version 5, and certainly not on a machine with less than 586MB of RAM. GUI, yes; graphics, no—gimme DESQview. (By the way, how does a little Commodore 64–based software firm—Geoworks Ensemble—do a better job on a multitasking GUI than mighty Microsoft? Could it be the combination of a good assembler and talent vs. C and an army of drudges?)

David Derbes
Chicago, IL

DISASSEMBLERS DISCUSSED

This is in response to Brett Glass's "Disassembler roundup" (*PJ* 9.2). I usually do not write letters to the editor, but this article struck a nerve.

I agreed with the author on several points, but it is obvious from the author's opening and closing statements that he does not like disassemblers. Disassemblers are tools, not mysterious, forbidding, magical programs. Anyone can use them. They can be used to learn assembly language and to advance one's programming techniques and abilities, or to reconstruct lost source code or the code from unsupported programs. In short, disassemblers can be put to whatever use the *user* wishes. Disassembly is as much a science as any other programming activity. It is not a programmer's last resort or a quick and dirty cure-all.

Although Glass's article was, for the most part, factual, it missed the essence of the four programs he reviewed. It conveyed no feeling for how the programs worked or what they looked like. I would not go out and buy one, nor could I recommend that someone else buy one, based solely on the information in Glass's article.

If Brett Glass does not like disassemblers, then he should not write articles about them.

Mark M. Janecki
Gilroy, CA

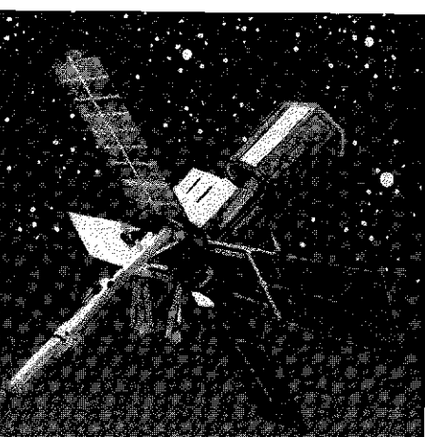
I want to commend *Programmer's Journal* and author Brett Glass on the "Disassembler roundup." I realize that this article was not intended to be a thorough review of disassemblers, but I do have a couple of items that *PJ* readers might be interested in.

The first is the use of Glass's TORTURE.ASM program as a test case. Today's disassemblers are fairly smart at separating code from data—historically, their weakest point. Code sequences that do not fit the "normal" patterns are suspected of being data rather than code. When you feed a disassembler intentionally strange code (such as TORTURE.ASM), you may

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The MetaWare Extended-DOS High C *Globally Optimizing Compiler* is for developing mission-critical applications that require:

- Compilation of large programs (one million+ lines)
- Ease of conversion of Microsoft "or other" code to 32-bits
- Support for the MetaWare 32-Bit Source-Level Debugger
- 100% ANSI Conformance
- Ease of migration to other development environments: DOS, UNIX System Vr3&4, SunOS, AIX, ...
- Global optimizations and Profiler for maximizing speed of code
- C++ as a future development language



```
if (environ == critical) {
    survival = High_C;
```

The MetaWare Extended-DOS High C *Locally Optimizing Compiler* extends the capability of our industry-proven technology by:

- Exploiting the Power of AutoCAD® ADS and DESQview/X™
- Ease of conversion of Microsoft "or other" code to 32-bits
- Providing an execution Profiler that helps developers increase execution speed by identifying bottlenecks
- Supporting the MetaWare 32-Bit Source-Level Debugger

Professional software developers live and breathe in challenging development environments unknown to most programmers. As a matter of survival, these Pioneers depend on the precision, performance, and portability that only MetaWare delivers. The superior reliability and functionality of our quality compilers and tools allow these Explorers to navigate unfamiliar development terrain with confidence.

But even the best tools are not enough. You need a close, professional relationship with the veteran team that pioneered Extended-DOS compilers way back in 1986; this technical support is the best "Life support" available today. MetaWare now extends your development horizons with two choices in compiler technology. Both *Locally* and *Globally* optimized technologies support the new MetaWare 32-bit Source-Level Debugger.

To learn more about how MetaWare can help you survive in your environment, or to place an order, call now.

MetaWare®
INCORPORATED

Compiler Products for Professional Software Developers

Circle 9 on Reader Service Card

2161 Delaware Avenue • Santa Cruz, CA • 95060-5706 • 408/429-6382 • FAX 408/429-9273

MetaWare, High C, and Professional Pascal are registered trademarks of MetaWare Incorporated. Other names are trademarks of their respective companies.
© Copyright 1991 MetaWare Incorporated

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

well find "incorrect" conversion of code to data and vice versa. Yet this is the proper result if the algorithms function correctly; disassemblers are designed to do their best jobs on the "normal," most common kinds of code.

A second—admittedly picky—point was the statement that MD86 did not "handle" addresses in segment:offset form. MD86 is very flexible in this regard, and almost any form of address input is acceptable. The manual (section 6.0) clearly spells this out. The user even has a choice of display formats.

Again, I wish to thank Brett Glass for taking the time to prepare this article. I hope we can look forward to a more in-depth review in the future. This subject has been kept in the dark too long.

Clark A. Calkins
Walnut Creek, CA

I use disassemblers as part of my job in writing very specialized compilers and have access to quite a few of these tools (far more than Brett Glass's review covered). Glass did not touch on some very important aspects.

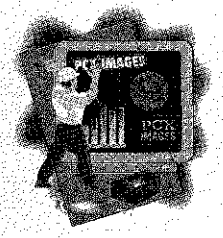
I use disassemblers for two main reasons: to view code generated by compilers (mine and others) to find ways to improve its efficiency, and to obtain a program's source when it has not been provided. This implies being able to edit, reassemble, and *execute*.

Each disassembler has strong and weak points, and having more than one has definite advantages. However, the fact that disassemblers differ in their treatment of immediate values that are so common in code (such as MOV AX,1234) bothers me. Although the immediate value may look like an ordinary constant from the disassembler's perspective, it may actually be either a constant (such as a loop counter) or the address (offset) of a variable or procedure.

This small point is vital to successful execution if you plan to edit and reassemble the resulting assembly code—even though your code may appear to assemble and link just fine. As an example, consider the following small program:

```
     org    100h
begin: push   cs
       pop    ds
       mov    ah,2ah    ;get current month into
                         register "dh"
       int    21h
       mov    cx,12     ;look up month name in
                         table
       mov    si,offset months
lp:    dec    dh
       jz     gotit
lpl:   lodsb
```

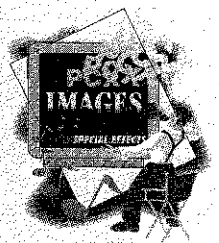

ADD THE POWER OF GRAPHICS TO YOUR PROGRAMS WITH GENUS



PCX PROGRAMMER'S TOOLKIT 4.0

Finally, a programmer's toolkit that helps you incorporate graphics into your programs quickly and easily. The PCX Programmer's Toolkit includes over 90 routines to display, save, scale, and print PCX bitmapped graphics from almost any program.

ONLY $249



GX EFFECTS 2.0* NEW!

Add spectacular special effects (F/X) to your programs, without being trapped in someone else's editor or slideshow program. Wipe, split, crush, slide, sand, drip, diagonal, spiral, random, or explode your graphics. Includes new animation support and sprite capabilities.

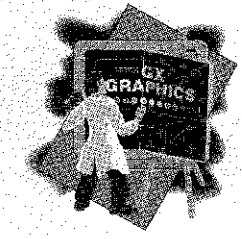
ONLY $199

* PCX Toolkit, GX Graphics, or GX Text recommended but not required.

GX TEXT 2.0 NEW!

Now you can display blazing bitmapped text in any graphics mode as simply as text mode. With GX Text, you can display your textual information right along with your graphical data, using the display or virtual buffers. Includes a graphical font editor and 1MB+ of fonts.

ONLY $149



GX GRAPHICS 1.0

A complete graphics library supporting all graphics primitives. Use the GX Graphics toolkit instead of the BGI or MS libraries and make your program faster, smaller, and more portable across compilers – while supporting more video modes.

ONLY $199

FOR THE GENUS GX DEVELOPMENT SERIES CATALOG, CALL TOLL-FREE:

1-800-227-0918

VISA, MASTERCARD, AMERICAN EXPRESS, CHECK, C.O.D., PURCHASE ORDER

G·E·N·U·S MICROPROGRAMMING

11315 MEADOW LAKE • HOUSTON, TX 77077 • (713) 870-0737
FAX (713) 870-0288 • BBS (713) 266-9362

Circle 10 on Reader Service Card

New!

Write c_ps.lib PostScript From C!

Now you can support PostScript devices through the C language!

c_ps.lib

- ✓ Graphics, Text, Line-Oriented Output
- ✓ Downloadable Font support
- ✓ ANSI, K&R, PC, UNIX, Mac

Only $195

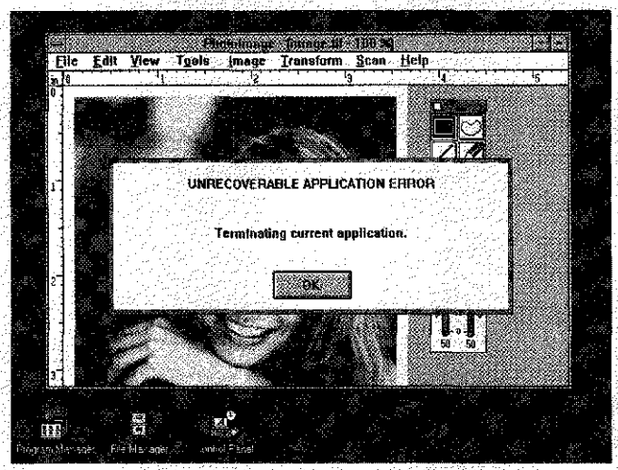
Barton Creek Software
2222 Western Trails # 106 Austin, TX 78745
(512) 441-8354
Fax (512) 444-4554

Circle 12 on Reader Service Card

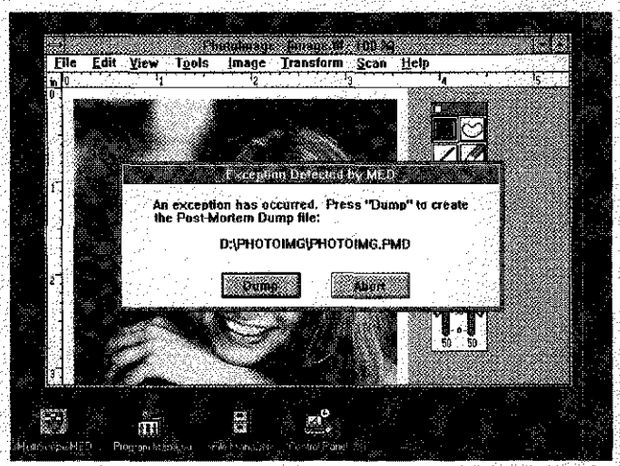
PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

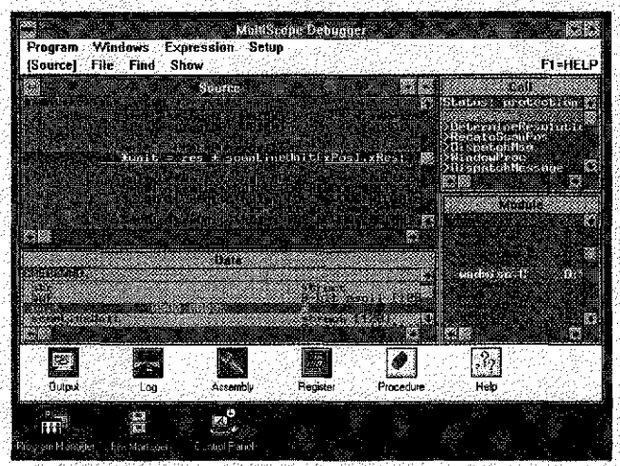
Experiencing "Unrecoverable Application Errors"?
Don't Fear...
MultiScope Crash Analyzer is Here!

Without the MultiScope Debuggers for Windows:



With the MultiScope MED (Monitor Execution and Dump) and Crash Analyzer tools:


File Edit View Tools Image Transform Scan Help
Exception Detected by MED
An exception has occurred. Press "Dump" to create the Post-Mortem Dump file:
D:\PHOTOIMG\PHOTOIMG.PMD
Dump
Abort



For more information, call MultiScope Sales at (800) 999-8846

MultiScope Debuggers for Windows provide the MultiScope Windows Run-Time Debugger, MED and Crash Analyzer for an introductory price of $379

MULTISCOPE

415 968 4892 PHONE
FAX 415 968 4622

Circle 61 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
        cmp     al,'$'
        jnz     lp1
        loop    lp
gotit:  mov     dx,si
        mov     ah,9      ;now display the month
        int     21h
        mov     ax,4c00h ;and exit
        int     21h
months:db       'January$'
        db      'February$'
        db      'March$'
        db      'April$'
        db      'May$'
        ..... etc.
        end     begin
```

I disassembled this with Sourcer, Dis•Doc, MasterKey, MD86, and Snoop. [Sample output is included on the code disk. —*Ed.*] I ran all programs as "automatically" as possible, with the least input from me. I could correct most "problems" by entering the proper instructions in every case, but I don't care to do that, especially on the longer (real) programs that I take apart.

All the disassemblers except MD86 generate the instruction **mov si,122h** at address 109 in place of the **mov si,offset months** instruction in the example source file. In fact, only MD86 seems to try to determine whether immediate values are offsets or numbers. Although Sourcer knows that the month-name table is referenced and includes a label for it, it does not use the label in the main program. Strange, no? Interestingly, both Dis•Doc and Snoop messed up the first part of the name-table area—and this is simple code.

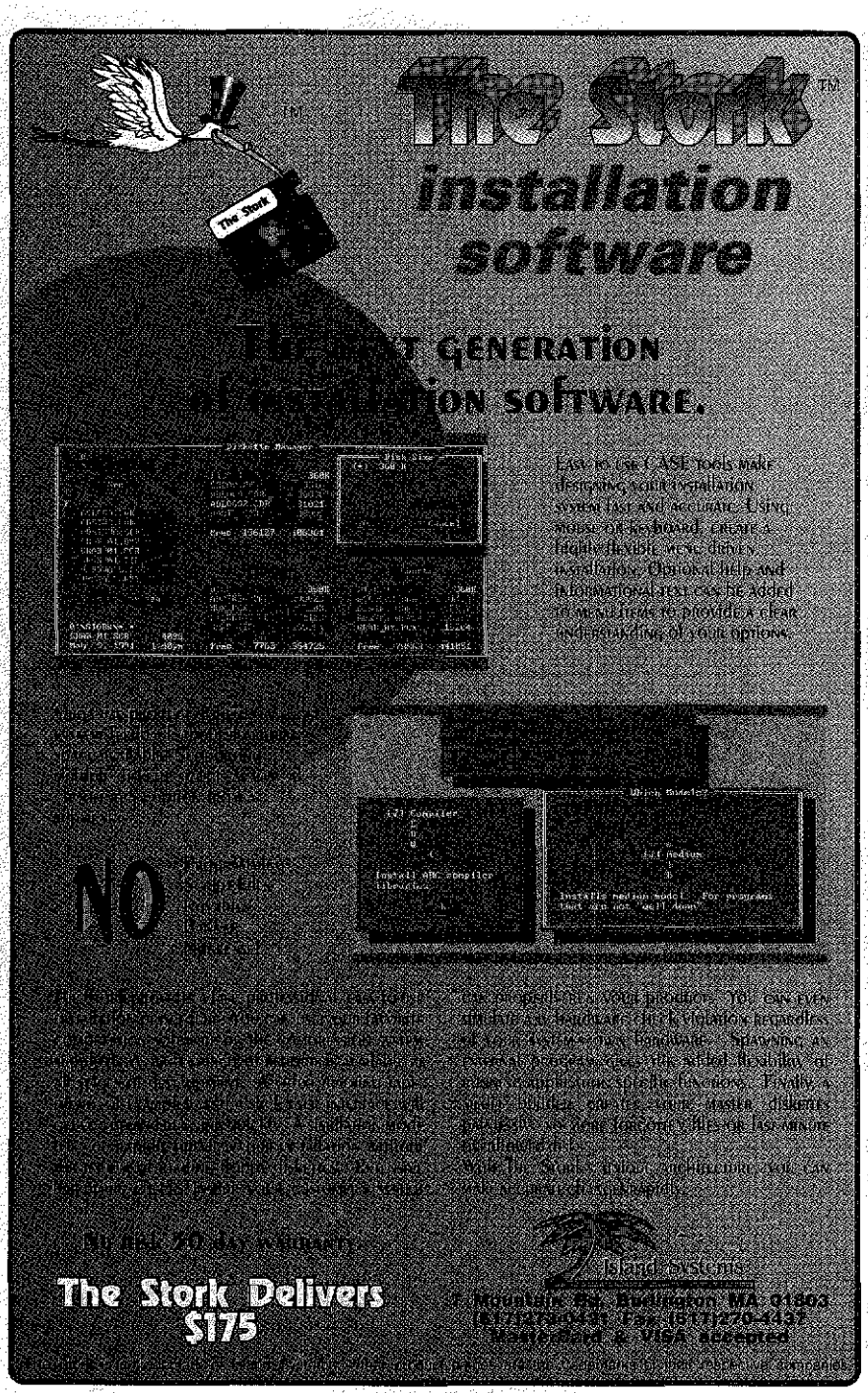
Each of these programs has good and bad points, and each user has pet peeves. My intent here is to point out that the review did not shed much light on these products that would be of help to a prospective purchaser. I hope that Brett's next review will delve more deeply into the products.

Joan Flugel
Sunol, CA

Thanks for Brett Glass's "Disassembler roundup" article. I do a lot of disassembly—some for the fun of it, and some to deal with undocumented code from defunct sources.

Glass is right that Sourcer is geologically slow; let it run overnight while you sleep. The huge listings it produces are nearly as incomprehensible as a raw hex dump; kilobytes of code become megabytes of listing.

I have had Sourcer for a couple of years, but I don't use it anymore. Instead, I use DEBUG, with redirection of input and output. Its simplicity is an advantage:



Circle 13 on Reader Service Card

Developer's Office Management Software
Turn Your Office Productivity Around!
Easy to use computerized system to organize, operate and control your software business. Written for DOS and DESQview API. The Developer's Assistant also works with Windows and Software Carousel.
Major features: | Other features | Affordable
User and Dealer indexed database
"Bug" look-up DB for software support
Superb invoicing capabilities
Utilities i.e., serial # embedding program
Directory of packagers, suppliers, etc.
Directory for product releases
Lots of reports and other features
Supports both laser and dotmatrix printers
Free 900 ☎ number to support your software
Built-in calculator, phone dialer, other accessories
Powerful • Integrated • Versatile • Slick
(512) 834-8927 $129.00 (+ tax in Texas)
(VISA/MasterCard/AMEX accepted)
S.L.A.M. Software
3202 W. Anderson Ln. Ste. 208-176
Austin, TX 78757-1022
Circle 11 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

DEBUG doesn't kibitz—it just does what you tell it to, and quickly.

John Sprung
Universal City, CA

GLASS'S MODEST PROPOSAL

I'm sorry that Mr. Janecki interpreted my remarks to mean that I dislike disassemblers. I do *like them—I'd just like to see them do their jobs better and faster.*

Two of the biggest stumbling blocks for any disassembler—including those I reviewed—are separating code from data and interpreting data structures. Most disassemblers are confused when the contents of a register vary on different passes through the same code, and even those that do register-level simulation often fail to predict the result of an address calculation. The four disassemblers I examined all had trouble determining whether a constant loaded into a register was an address, an offset from a base address, or a value whose function was strictly mathematical.

What is the best way to differentiate code from data? How can one know exactly how the parts of a program work when the code is actually running? I'd like to stick my neck out one more time on this issue and propose a new approach.

Intel's 80386 and 80486 microprocessors can run programs in the virtual 8086 mode, in which every instruction is subject to scrutiny by another program that monitors it and controls the machine. What if one were to construct a monitor program that snooped on an application as it was running—noting where the instructions were and what the registers contained at key points—and fed that information back to the disassembler? If the user exercised even 30% of the program during such a run, the hidden monitor could learn enough about the program to make the test easily decipherable.

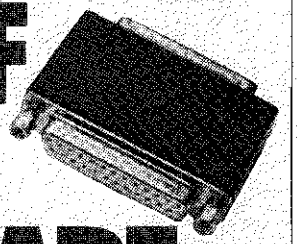
Creating such a monitor/disassembler combination wouldn't be easy. Virtual 8086 mode is tricky and is not well documented. An inefficient monitor program could slow the application down intolerably, making it difficult to run test cases. And if the application used EMS, the monitor might have to emulate this function to exercise the expanded memory code. Still, the ability to decipher programs would be well worth the effort.

An ideal disassembler would be able to figure out everything about a program with little or no user input. Although none of the reviewed disassemblers has this capability, this proposal might help to make such products possible in the future.

Brett Glass
Palo Alto, CA

ERRATUM: A wrong telephone number for MetaWare Inc. appeared in "Impressions," *PJ* 9.3, p. 78. The correct number is (408) 429-6382.

DON'T LOSE 1/2 OF YOUR SOFTWARE PROFITS!



FOR FURTHER INFORMATION, CALL TODAY! 1(800) 456-2587

Save your lost profits with SecuriKey — a unique hardware device that prevents unauthorized use and duplication of computer software.

- IBM, compatible & Macintosh
- Allows unlimited backup copies
- Does not permit duplication
- Can record software usage
- Available in custom colors & logos
- Variety of programmable options
- UNIX/XENIX

micro security systems, inc.
4750 Wiley Post Way, Suite 180
Salt Lake City, Utah 84116
FAX (801) 575-6621

Circle 14 on Reader Service Card

"Without a doubt, OPUS Make is the hottest make utility on the market"
Tom Swan, PCWorld

OPUS SOFTWARE

OPUS MAKE

For professional programmers, OPUS Make is the superior choice in a make utility. It is the fastest most full-featured make utility there is. Features include:

- DOS version uses only 3K memory!
- Multiple directory support.
- Supports: Polytron PVCS™, Burton Systems TLIB™, Microsoft LIB™, SLR Systems OPTLIB™
- Generates automatic response files of unlimited length for LINK and LIB.
- OPUS MKMF included: An automatic dependency generator that fully understands C preprocessor directives.

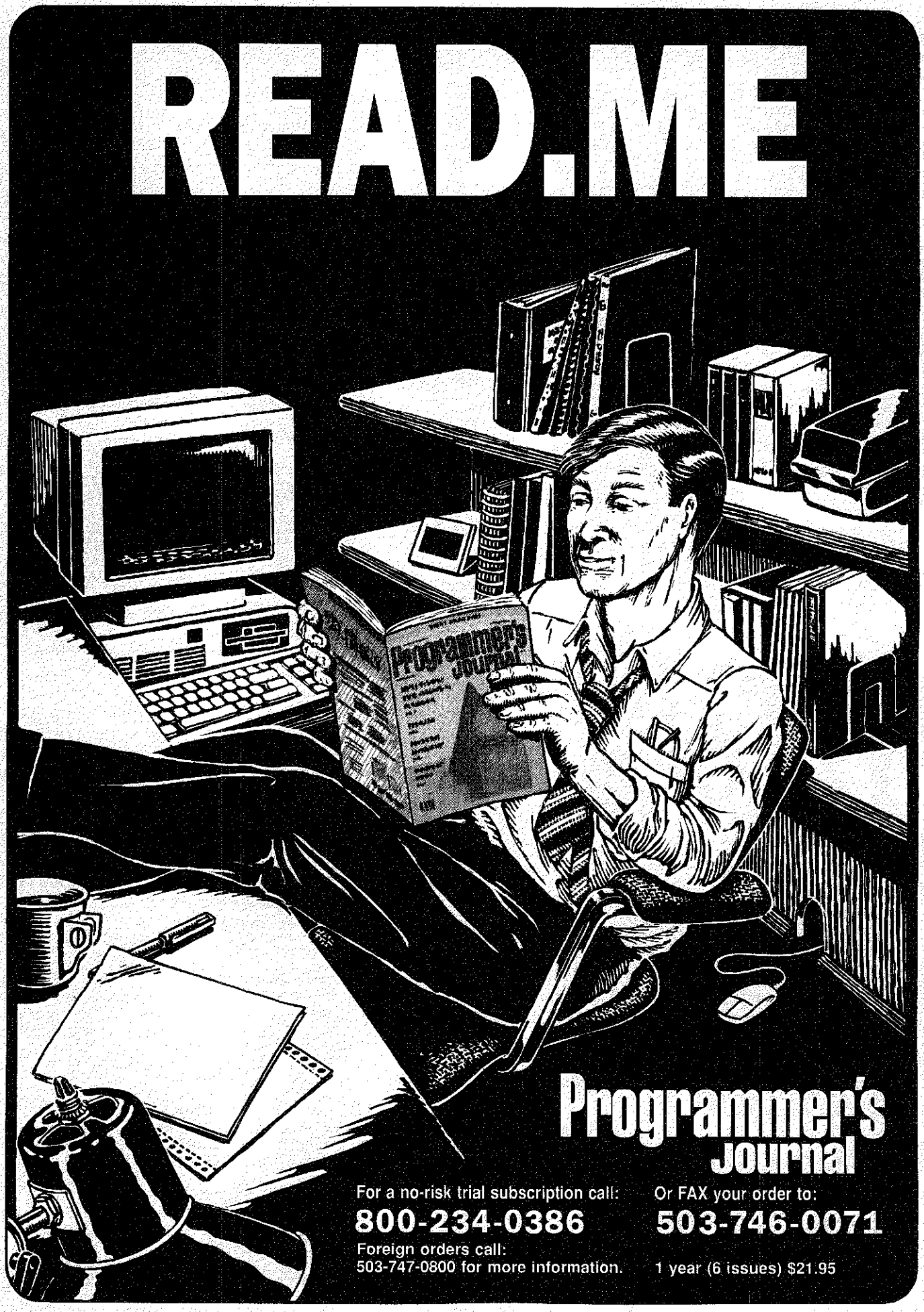
Both DOS and OS/2 executables for only: $129 MC, Visa, COD, or PO. 60 day MBG!

Call 1-800-248-OPUS
International: 415-964-1901
1032 Irving Street, Suite 439, San Francisco, CA 94122
Trademarks owned by their respective companies

Circle 15 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

READ.ME



Programmer's Journal

For a no-risk trial subscription call:
800-234-0386
Foreign orders call:
503-747-0800 for more information.

Or FAX your order to:
503-746-0071
1 year (6 issues) $21.95

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Craftsmanship Isn't a Lost Art

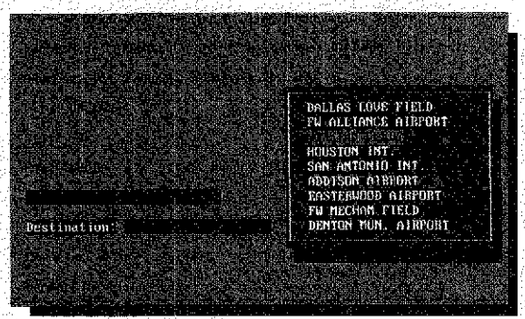
Enjoy a well-crafted user interface for C - one with all the details you need

Like the gears in a skillfully crafted timepiece, the low-level functions of Greenleaf DataWindows mesh and work in concert, hiding the complex details, to make your interface design easier!

Making an overlying window invisible and repainting the screen takes a single function call with DataWindows. Autodetection of display type, Enhanced Keyboard and mouse has never been easier. We do the grunt work so you don't have to!

Select from several styles of menus, including Pull-Down and Pop-Up, with infinite branching and runtime extensibility, 3 kinds of help and hundreds of other features!

Best of all a sophisticated data entry engine handles your database record as an entire transaction (unless all fields check out, the record isn't updated). Scrollable subforms, memo and tickertape fields, nested transactions, choice boxes, list boxes, 19 kinds of state-change function hooks for programmability - and so much more!



Forms larger than the display handled easily with auto or manual scrolling any direction.

Data entry and menus are based on logical windows which are special in DataWindows. The specialness comes from extra care at the lowest levels - so that you don't have to concern yourself with details like updating the screen! Save windows with text & attributes to disk.

User input from keyboard and mouse is monitored by a single call. Extras like input redirection to a menu or string, Ctrl-Break handling, and automatic display type sensing are built in.

Documentation includes two binders chock-full of data, tutorials, and compilable examples of each function, PLUS a complete online documentation system including a FREE PDQPlus™ engine! Other help engines supported too! A Greenleaf Exclusive!

Portability is assured by versions for DOS, UNIX, XENIX and VMS (including terminal definitions for multi-user systems).

So easy to use - yet jam-packed with extras. Used by professionals the world over.

CALL (800)523-9830 *UNIX, DOS, 386!*

FAX (214)248-7830

Greenleaf DataWindows v2.12 for DOS supports Microsoft C, Borland/Turbo C/C++, Watcom C, Zortech C++ and TopSpeed C . Versions for UNIX System V and SCO XENIX System V.

- ***Greenleaf MakeForm v2.0 Form Generator also available!***
- ***No royalties.***
- ***Source code is FREE!***
- ***Support is FREE!***
- ***Multi-User Extended Support is also available.***
- ***Free BBS and Newsletter!***
- ***Ask about Gold Support!***

GREENLEAF Software

16479 Dallas Parkway, #570
Dallas, TX 75248
(214)248-2561

Member TopSpeed® Consortium

NEW PRODUCT!

DataWindows/386™

- ✓ Supports Phar Lap 386 DOS Extender
- ✓ Uses Watcom C 386 v8.0 or MetaWare High C v2.3
- ✓ All source code in ANSI C supplied FREE!
- ✓ Supports full 32-bit flat model for 386 or 486 systems.
- ✓ Logical windows — up to available memory — AND you never have to worry about repainting because DataWindows/386™ does that automatically — always!
- ✓ All features of DataWindows v2.12 (excepting DOS Redirection).
- ✓ Fully compatible with Greenleaf MakeForm v2.0 form generator.

Circle 17 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Pink noise

By Frank D. Greco

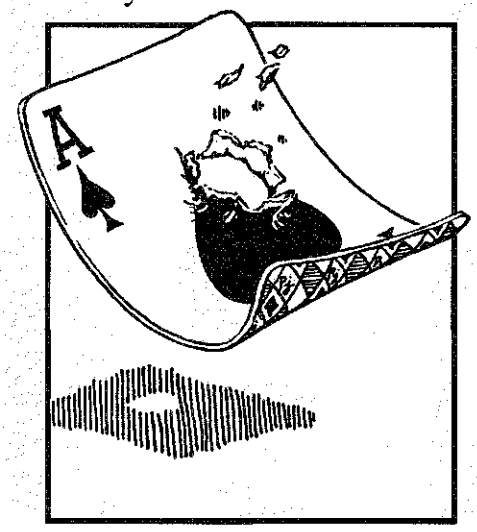


ILLUSTRATION BY JIM CARPENTER

The recently formed ACE (Advanced Computing Environment) group—which combines Compaq, Microsoft, Silicon Graphics, MIPS, DEC, and others—is a collaboration of companies that have not made it in the workstation field. Compaq wants to escape the Intel jail. Microsoft is still trying to force-feed us OS/2 (MS wants you to call it "NT" so you won't think it's the failed OS/2). Silicon Graphics wants desperately to be a big-time workstation vendor (and needs money to do so). MIPS is still pushing their chip as a standard (a standard that is controlled by the MIPS corporation—unlike SPARC, which is controlled by SPARC International, not Sun). And DEC is trying anything to buy time for their own proprietary chip, which won't be ready for another two years. (My DEC prognostication: "swing and a miss, strike three.")

I've never seen such a motley crew of workstation hopefuls. Within this aggregate are five camps of operating system standards (SVR4 Unix, Ultrix, OS/2, OSF/1, and DOS), two processor standards (Intel and MIPS), three standard bus architectures (DEC, VME, and EISA), three user interface standards (Windows, Motif, and Presentation Manager), two byte-ordering standards (big-endian Silicon Graphics style and little-endian PC style à la DEC and Microsoft) and two graphics standards (Silicon Graphics and PEX).

This is a standard environment? Who's kidding whom?

MIPS CHIP SLIP

What's the real reason behind Compaq's choosing the MIPS chip? Compaq CEO Rod Canion was quoted as saying that Compaq wanted a high-performance, multiple-source chip whose manufacturer does not build systems and compete with its customers, like Sun or Intel does. Somebody should tell Mr. Canion that MIPS has been selling a complete line of "Magnum" workstations and file servers based on the company's own chip. Do you think that MIPS will stop making their own workstations and higher-profit-margin servers because Compaq is one of their OEMs? Not likely.

To succeed in the high-performance workstation market, Compaq must start with a commodity chip like the SPARC. Having a small piece of a large SPARC market is much better than having a large piece of a nonexistent MIPS market (MIPS has posted two financially disappointing quarters in a row). And if they think that having Microsoft's OS/2 on their side is going to help them, Bill Gates is better at selling snake oil than is DEC's CEO, Ken Olsen.

The acquisition of 13% of Silicon Graphics will benefit Compaq's Intel-based PCs more than anything. An MS-DOS-based Compaq 486/586 box with sexy SGI graphics will make many a CAD/CAM user drool. (SGI has been trying to sell their PC graphics boards for a long time; they have virtually no distribution channel.) Unfortunately, IBM also owns a substantial chunk of SGI and uses some of this graphics technology in their RS/6000 workstation. IBM is also rumored to use SGI technology in their PS/2 machines. The IBM connection significantly offsets the competitive advantage of Compaq's minority SGI investment.

I think Compaq chose the MIPS chip because they wanted to be seen as a market leader. But now they've become part of an amorphous pseudo-standards-setting committee (along with such industry "stalwarts" as DEC, Unisys, Wang, and Control Data Corp.) that's agreeing to use OS/2 as part of a technology that won't be ready for at least two years (by which time Sun will have a huge lead).

Compaq's expertise in packaging systems and their tremendous advantage with the VAR channels would have given Sun a run for their money; it seems logical for Compaq to have chosen SPARC. Did Microsoft swing the Compaq cloners away from the SPARC camp? Mean Green knows that controlling the operating system and user interface is a tremendous strategic advantage. If the world went SPARC and Unix, Gates's group would lose its edge.

X-RATED WINDOWS

First the *Wall Street Journal* reported that Microsoft had abandoned OS/2 development and even quoted some Microsoft executives as saying that MS would be concentrating on Windows for future products.

The next day, presumably after someone at IBM read the paper, Microsoft issued a statement that identified IBM as doing the near-term OS/2 and Microsoft as doing the far-term OS/2 (aka NT). I guess MS forgot that IBM is still a very large customer of theirs. Despite the correction, the truth is out: Mean Green will concentrate on an architecture-independent Windows, not on OS/2.

Microsoft knows that Windows represents large licensing profits and a huge business advantage for their applications division. Windows over a RISC platform would even give them a foot (or at least a toe) in the workstation door. It seems that the PM/X (Presentation Manager

Frank D. Greco is an independent consultant who provides industry analysis, systems consulting, and technological forecasting for Fortune 500 companies. Interested readers can contact Frank c/o Crossroads Technologies, P.O. Box 530, Fanwood, New Jersey 07023.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

over X Windows) project within Microsoft is still alive, but with Presentation Manager DOA, it's now MS Windows over X.

THE S-BUS STOPS HERE

If you want to run high-performance PC programs on your SPARCstation, you can get 80386 and 80486 S-Bus boards from a variety of vendors. As rumored in *Digital News*, Sun will shortly announce a 486 to hang off the SPARCstation's S-Bus, so you can run your favorite MS-DOS programs in an OpenWindows window.

POINTER POINTERS

Proxima Corporation has some high-quality LCD projection systems for PCs and Macintoshes. They recently announced a wireless pointing device that can be used on a projected image. A camera mounted on the projection panel actually reads the pointer's light off the projected image. With this device, you no longer need another person operating the computer while you speak, making it the perfect choice for one-person presentations.

DRIVING MISS DAISY

Toshiba, Prairietek, and Areal Technology have shown single-platter, 2.5-inch disk drives that are 0.75 inches tall and weigh 6 ounces. They range in capacity from 20 to 150 MB. These drives are perfect for notebook computers. It's only a matter of time before multiplatter versions with capacities of more than 300 MB appear.

Also expect to see 1-gigabyte, 3.5-inch drives at next year's Comdex.

LAPSE OF LUXURY

Toshiba is selling their SPARC laptop only in Japan because of an agreement with Sun. This agreement will run out in late 1991 or early 1992. According to Toshiba sales representatives, the company is scheduled to roll out a series of SPARC workstations and laptops for the American market.

SUNS AND SUDS

I almost fell out of my chair while watching ESPN in late April. After a report on the Foreman vs. Holyfield boxing match, some coverage of the Masters golf tournament, and a bit on overpaid baseball players, what should my wondering eyes behold but a commercial for Sun Microsystems. It's incongruous to see a Sun spot on the tube following a message from Miller Lite.

ECLIPSED BY SPARC

It looks as though Intergraph will eschew their proprietary Clipper RISC chip and use the SPARC in future workstations. Intergraph, which does a huge business with the government and large electronic design companies, struck a sizeable OEM deal with Sun. All of the company's software will be ported over to SPARC-based machines. Clipper-based

The Programmer's Corner
— A Source Code Repository —

If you are reading this magazine, then you appreciate the art of programming. One of the most important tools for a software craftsman is example code. To meet this need, we provide more **than 9000 files** (over one gigabyte) of shareware and public domain source code and programs. We cover all popular languages including C, Pascal, and Basic. TPC even has hard to find source code such as OS/2 PM C source, C++ source, and Turbo Pascal 5.5 OOP source. Source code from this issue of *Programmer's Journal* is also available. New code is added daily, all in the interest of helping the professional programmer.

So call today: (301) 596-1180 300-9600 Baud
(301) 995-3744 24 Hours-A-Day
(301) 621-4636
Only $50 for a one year subscription

— TPC uses FastComm 9600 V.32 Modems —

TPC
The Programmer's Corner
5430 F Lynx Lane, Suite 224
Columbia, MD 21044

Circle 18 on Reader Service Card

Data Junction® converts data files!

Any **ASCII • dBASE • R:BASE • Lotus 1-2-3**
Excel • Paradox • Informix • Q&A • Oracle • Clarion
DataFlex • askSam • SC4 • Xdb • DataEase • Symphony
ACT! • Platinum • DAC • ACCPAC+ and other apps
Btrieve • c-tree • EBCDIC • Packed & Zoned decimal
IBM/Mips/Sparc/Cray/Vax/Intel/68000 Binaries & Floats

Filter Your Files While Converting!
- *Menu driven — No programming*
- *Select & rearrange records/fields/chars*
- *Instant scripting for batch converts*
- *Written in C — Ideal for OEM apps*

Clean Up & Massage Your Data!
- *Merge & split fields; Name/address parsing*
- *Search/replace and case translation by column*
- *Edit/enter corrections — Build custom data types*

1990 DATA BASED ADVISOR The Database Management Systems Magazine — Readers Choice Award

"... an invaluable integration tool." —*Byte*
"... more than worth its price." —*PC Magazine*
"... a surefire winner." —*Computer Language*

Standard $99— Converts dBASE, ASCII, 1-2-3, Q&A, Enable, DIF, askSam, & merge files for WordPerfect, Word & Wordstar
Professional $199— Standard formats plus Binary/EBCDIC, SC4, SYLK, Paradox, R:BASE, Clarion, Excel, DataEase, DataFlex, c-tree, Btrieve
Advanced $299— Standard & Prof. formats + Informix, Xdb, Oracle, C-ISAM

Call **800/444-1945** or 512/482-0824 • FAX 512/482-0976
Also available for LANs, Xenix & Unix. MC/VISA • AMEX • COD • Corp. PO.
Tools & Techniques Inc • 1620 W 12 • Austin TX 78703

Circle 19 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Intergraph workstations hold 4% of the worldwide workstation market.

NICE THREADS

At a recent AT&T System V Release 4 conference at the Bell Laboratories in New Jersey, AT&T's technical rep stated that they were not putting lightweight processes (LWP) in SVR4. I was told that their implementation of **fork** (the Unix system call that creates a new process) is so efficient that SVR4 does not need LWPs.

AT&T doesn't realize that the biggest advantage of LWPs is that the processes (actually, multiple threads of a single process) share the same data space; heavyweight processes do not. You still need expensive Unix IPC to communicate among heavyweight processes, offsetting any performance increase from their faster forking.

With LWPs, if a thread needs some data from another thread, you just use it! Of course, if more than one thread needs a common data structure, you still have to coordinate access using some sort of monitor (semaphore) mechanism.

Despite its move toward SVR4, SunOS will also accommodate multithreading in their version of the OS. I imagine that, like Mach, multithreadedness will find its way into future versions of the Unix kernel.

THERE'S JOY IN MUDVILLE

A few years back, Sun head techno-nerd and industry personality Bill Joy said that Sun would go after the graphics market. The GX/GXP graphics co-processors and 30-MIP SPARCstation 2 workstations were initial steps toward replacing Silicon Graphics as the king of the graphics-workstation hill.

Now Sun has taken another large step toward graphics dominance by acquiring the technology, patents, and principals of Sutherland, Sproull, and Associates (SSA). This acquisition is significant: SSA has some very advanced graphics technology. They have built extremely realistic real-time flight simulators for the military. Back in the 1960s, Dr. Sutherland built the first computer program that could be called direct-manipulation (a MACpaint-type program).

Dr. Sutherland and Dr. Sproull—both former Xerox PARC members—will be joining Sun Laboratories, which concentrates on commercializing technology for future Sun projects.

FEATS OF CRAY

Cray is investigating a future supercomputer that can also run commodity software. They are evaluating whether to use SPARC or MIPS chips. Given the many Sun-to-Cray bridges out there in academia (such as Sun/Cray Hyperchannel connections, Sun graphical front-ends to Cray floating-point power, etc.), the smart money is on SPARC.

SURREAL SCREEN TEST

It looks as if the standard screen for portable computers will be small. The trend

INSTALL.BAT

COMPUTER LANGUAGE PRODUCTIVITY AWARD 1990

First impressions last.

INSTALL 3.0 gives your software product a professional introduction.

Installations that rely on batch files look amateurish and have limited error handling. INSTALL 3.0 creates an elegant installation procedure that will increase user's confidence in your software product.

INSTALL 3.0 comes with many samples you can modify and use immediately; no programming usually needed. All available RAM is used for fast file transfers.

INSTALL 3.0 detects monitors, DOS versions, coprocessors, etc. to create machine-specific installations without asking questions.

- Features very sophisticated error-handling
- Does high performance data compression
- Can safely and reliably modify system files
- Allows selective installation of parts of your program
- Complete source code included
- No royalty charges
- Free technical support

INSTALL 3.0 has been used for over four years by some of the biggest (and smallest!) names in the industry, in the U.S. and abroad, to install millions of copies of programs. Make sure your software product makes a good first impression with INSTALL 3.0.

KNOWLEDGE DYNAMICS CORP.
Highway Contract 4, Box 185-H
Canyon Lake, TX 78133-3508
MasterCard/VISA/COD/POs welcome

$249.95 + shipping
Money back guarantee
Sales 800-331-2783 Ext 0196
(internat'l 512-964-3994
24 hr FAX 512-964-3958
24 hr BBS 512-964-3929

Circle 20 on Reader Service Card

We Have Ways To Improve Your Productivity

TE Developer's Kit

Why reinvent the wheel? TE Developer's Kit provides the most cost effective way of incorporating text editing features into your application. The kit includes the complete source code for the editor and the executable file. Supports Microsoft and Borland 'C' compilers.

Features: multiple files/windows, word-wrap and printing, edits files larger than available memory, reconfigurable keyboard commands and screen colors, undo, column block commands, and search/replace. The kit also includes TER small editor routine. TER is a subset of TE. Your application calls this routine by specifying the window location and size, maximum file size, and an input buffer or file. *(DOS: $169, Windows: $269)*

WinMem: Windows 3 Memory Manager

WinMem provides an unlimited amount of virtual memory even in Windows' Real and Standard modes. The device independence is not enough. WinMem offers you CPU and memory resource independence. It provides an unlimited number of global memory handles, and reduces the overhead per block to 4 bytes (from 24 bytes of Windows). *($139)*

Spell Time *(NEW)*

Spell Time consists of a dictionary (over 100K words) and the routines to access the dictionary. Also includes an application specific dictionary, and a user dictionary. The routine will suggest alternates for a misspelled word. Highly optimized: 50 to 75 words/sec on a 12 Mhz 286 computer. Uses virtual memory on disk or EMS. An interface with TE included. Includes the complete source code. *($369)*

Sub Systems, Inc. 1-800-447-6819
Fax: 1-617-438-0311
159 Main Street, #8C, Stoneham, MA 02180, 1-(617)-438-8901

Circle 21 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

in notebook computing is for screens no larger than an 8½ by 11 piece of paper.

How about using virtual screens? Military jet pilots use this technology (called a "heads-up" display). A screen is displayed in virtual space in front of the pilot, who wears special goggles. A scaled-down version for PCs is available, but it's limited to two colors and CGA resolution. A heads-up display for portable workstations would come closer to the military version, wrapping a large virtual monitor around the user. The possibilities being offered by virtual reality proponents are even more hallucinogenic. My suggestion is tame by comparison.

IS A WARP DRIVE NEXT?

Intel appears to be working on a transputer-like processor called the iWARP chip. Rumor has it that Intel recently demonstrated a machine that contained 128 iWARP processors capable of supercomputer speed.

EYES ON TEXAS

Based on talks given recently by Sun executives Wayne Rosing and Andreas Bechtolsheim, I expect to see superscalar 60–100 MIPS SPARC workstations by the end of the year—with a generation jump (above 100 MFLOPS) in floating-point performance.

Judging from recent articles in technical journals, I expect that this upcoming machine will be all Texas-made, featuring the TI Viking SPARC chip and a gang of TI 390C602A floating-point chips.

SHOWING SOME GRAY

Look for Sun to move completely away from monochrome monitors and to switch to minimally 8-bit gray-scale monitors.

Also, expect them to use anti-aliasing for font rendering—the GX graphics accelerator, which is now standard on all SPARCstations, is capable of performing anti-aliasing in hardware. PD

Resource Converter

Translate your resource files from **PM to Windows,** or **Windows to PM**. Save weeks of trial and error converting them yourself. Get the job done right the first time with the **Resource Converter!**

Converts:

Dialog Templates
All Control types
Std. Control ID's
Menus & Tables

PM to Win converter now shipping for $225, Win to PM shipping soon.

Petrasoft
3665 East Bay Drive Suite 204-112
Largo, Florida 34641
Msg: 813-530-9481, FAX: 813-530-9264

All trademarks or tradenames used here-in are owned by their respective companies.

Circle 23 on Reader Service Card

Tired of the CONFIG.SYS shuffle?

Let BOOTCON simplify your life.

"... it's bulletproof ... The interface is well thought out. You need this program."
Dr. Jerry Pournelle, *Byte*, July 1990

"Highly recommended."
John C. Dvorak, *PC Magazine*, July 1990

How many CONFIG.SYS and AUTOEXEC.BAT combinations do you have on your PC? BOOTCON eliminates the confusion by storing all your configuration information in just one set of configuration files. *Each time you boot your PC* you can select the configuration you want to use from a menu of up to 26 choices.

BOOTCON is ideal when you need to use conflicting programs such as Windows 3.0, QEMM/386, 386MAX, Soft-ICE, and Lotus 123. A default configuration will automatically be used if you do not make a selection within the specified time-out period. BOOTCON also provides an optional password protection feature.

BOOTCON lists for $59.95 and comes on both 5¼" and 3½" disks. It has an unconditional 30-day money-back guarantee. MS-DOS or PC-DOS 3.10 or later is required.

Modular Software Systems

115 West California Blvd., Suite 113
Pasadena, CA 91105

Phone (818) 440-9104 FAX (818) 440-9240

Windows and MS-DOS are trademarks of Microsoft Corporation. PC-DOS is a trademark of International Business Machines Corporation. QEMM/386 is a trademark of Quarterdeck Office Systems. 386MAX is a trademark of Qualitas Corporation. Soft-ICE is a trademark of Nu-Mega Technologies. Lotus and 123 are trademarks of Lotus Development Corporation.

Circle 22 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Don't be wrong about your rights

By William H. Roetzheim

At one point in my life I thought it would be fun to be a lawyer—or a fighter pilot or a computer programmer. I've had the opportunity to work as both a fighter pilot and a computer programmer, and I've outgrown my desire to be a lawyer. It seems that the most useful thing about lawyers is that they serve well as the brunt of lawyer jokes.

However, lawyers are definitely out there, and the long arm of the law is now invading the previously overlooked field of computer science. Taking some basic steps and thinking ahead can help you protect your rights from competitors and avoid possible lawsuits. As I said, I'm not a lawyer, and I'm not offering legal advice on matters such as filing for trademark protection, the specifics of which are covered in numerous books. Instead, I want to share some pointers on the types of things you should be aware of and might want to look into further.

ARE YOU TAKING NOTES?

You should be keeping an engineering notebook, where every idea you ever have gets written down, along with the date you have the idea. As you refine it (or add new facets), enter these modifications as well.

A notebook is a good idea even without lawyers—it allows you to keep track of ideas until you're able to pursue them. The law enters the picture under circumstances such as the following: suppose you develop a product based on an idea you had a long time ago, and someone claims that the product infringes on their patent. If you had the idea first, you have a valid argument against the attempt to enforce the patent.

Or perhaps you have a good idea that you begin to develop into a commercial

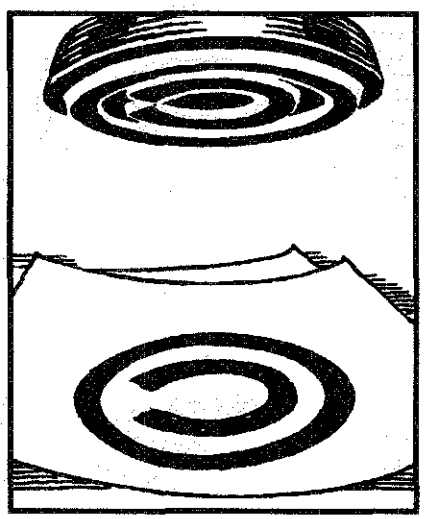


ILLUSTRATION BY JIM CARPENTER

success. Sometime later, a friend or co-worker tells you about their great new idea, which happens to be identical to yours. If the ex-friend or co-worker sues you for stealing, your engineering notebook will offer evidence that you had the idea independently.

Because your engineering notebook (or idea book) might be used as legal evidence, its format should follow its function. It should be bound, and each page should be numbered so that a missing or added page would be obvious. Blank books like this are available at many office supply stores and college bookstores.

Each entry in the notebook should be dated. Fill each page completely or draw a line through the unused portion of a page and initial it. You can have multiple entries per page; each page needs to be filled completely to avoid any suspicion that you added or modified an entry at a later date.

All entries should be in pen. You should periodically have all completed pages initialed by someone else. Because they are only verifying that the pages are filled, they do not need to read them.

GET YOUR COPYRIGHT RIGHT

Your software (program, listings, manuals, etc.) can and should be copyrighted. This ensures that anyone who looks at the item knows it is copyrighted. A copyright notice looks something like this: "Copyright (c) 1991, William H. Roetzheim." The notice must include the date and your name. If you leave off the "(c)," your copyright notice is valid in the United States but may be invalid internationally. To protect your rights internationally, you must use either the (c) or the actual symbol, ©.

Usually, you do not need to register your copyright until the product is complete. Even then, there is no absolute requirement that you formally register the copyright, although you have more rights if you do.

I like to have the copyright notice in three different places. First, I put it in the header block for each function. I've made the copyright notice part of my header template, so I don't need to think about it. Second, I store the copyright notice as a static data structure and display it on the title screen of the application. This shows the copyright to the user and ensures that it is embedded in the **exe** file. Third, I put the copyright notice in the program's manual.

BREATHTAKING TRADEMARKS

You may want to file for trademark protection for the name of your program and for any catchy names you use for functions or features. A trademark prevents a competitor from copying your name outright.

A trademark can be a marketing tool as well. Do you remember when Certs was the only breath mint to contain a "sparkling drop of retsin"? The manufacturer simply made up and trademarked the name "retsin" for the vegetable oil in Certs. Other manufacturers could say their product contained vegetable oil, but not "retsin."

CAN YOU KEEP A SECRET?

For many programs, the best protection for your source code is trade secret protection. As long as you take reasonable actions to treat the source code as a secret, trade secret protection is automatic. For example, on each page of your listings, print a proprietary data notice

William H. Roetzheim is a senior associate with Booz, Allen & Hamilton, and a noted author and speaker on software engineering topics. Bill can be contacted at Booz, Allen & Hamilton, Inc., Suite 109, 4025 Hancock Street, San Diego, CA 92110; (619) 223-5681.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

("This data is proprietary to XYZ Company"). Source code disks should contain a proprietary notice and should be numbered and tracked. Be careful about allowing access to your source code; ask nonemployees to sign a nondisclosure statement. Unwanted listings should be burned or shredded, not tossed out. Dispose of defective hard disks carefully—their platters still contain all your data.

PATENTS PENDING

Patenting everything in software seems to be the current rage, but I don't think it's a good idea for either you or the industry. This patenting mania stifles innovation, and a large percentage of the patents currently being awarded are not going to stand up under the scrutiny of the courts. The patent office is simply not qualified to judge these patents very well.

Nevertheless, you might want to consider patenting *portions* of your software. If you develop a unique, innovative algorithm or a significant improvement to an existing one, you should consider patenting both the algorithm and its implementing code. Keep in mind, however, that the patent process takes years. Will your innovation be OBE (overcome by events) by the time a patent is awarded? Patent protection is also much more expensive than anything else I am describing in this column, so you should consider these drawbacks before deciding to apply.

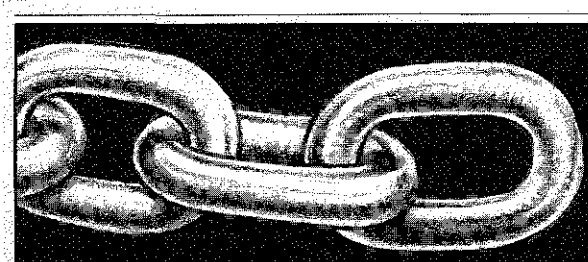
Big companies pay lawyers lots of money to write carefully crafted license agreements and limited warranties. They work hard to avoid any legal pitfalls or loopholes. Read the license agreements and warranties of some of the majors, identifying those clauses that sound good to you. You can then use these boilerplate sections as templates for preparing your own license agreement and warranty.

PRACTICE SAFE SELLING

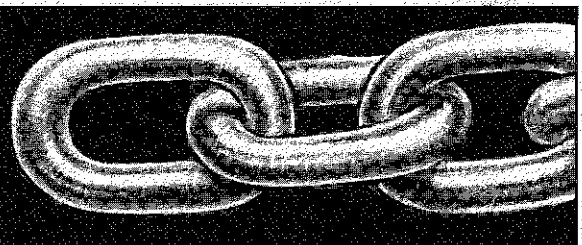
If you sell the rights to your software to a publisher, ensure that your contract includes a clause stating that the publisher assumes all liability for any suits filed in connection with the software. This includes things such as look-and-feel suits. In my case, I accept liability only if I am shown to have out-and-out copied the code.

If you decide to peddle your software on your own, I highly recommend that you consider incorporating first. This shields your personal assets (your car, house, and so on) from liability if the software company is sued. Most states have information on how to incorporate your business on your own. On a do-it-yourself basis, the process is relatively cheap (normally a few hundred dollars).

Take a few hours to begin implementing these ideas—then get back to your coding. The time you spend setting up procedures to protect yourself in the event of legal problems may very well save you thousands of dollars if push ever comes to shove.



THE MISSING LINKER!



NEW OPTLINK/WINDOWS: THE WORLD'S FASTEST WINDOWS LINKER.

Using a Windows linker isn't tedious and time-consuming anymore. With new OPTLINK/Windows 2.50, it's a snap!

OPTLINK/Windows is the fastest, most powerful linker available for Microsoft Windows. It performs up to 20 times faster than MS-LINK, generating DOS, Windows and OS/2 .EXE files.

Power Without Limits.

Blazing speed is only the beginning. OPTLINK/Windows is the only linker that sets *no* internal limits on the number of modules, libraries, publics or segments you link. OPTLINK/Windows supports CodeView 3.0 & 3.1, and lets you link and debug larger applications than any other linker.

Plus, OPTLINK/Windows utilizes EMS and XMS, and works wonders on a Novell network.

Look At What You're Missing.

OPTLINK/Windows is more than fast and powerful. It's packed with features and functions you can't find anywhere else:

- Copies .RES files to .EXE at link time
- Generates complete global cross-references
- Generates smaller files than MS-LINK
- Detailed segment mapping
- Compatible with all compilers using Intel/Microsoft .OBJ files

Don't Miss Out! Order Now!

OPTLINK/Windows 2.50 is priced at only $350. And it comes with a 30-day money-back guarantee.

Why waste your time with slow, stone-age linkers? Order OPTLINK/Windows now, and get what you've been missing!

SLR

SLR SYSTEMS, INC.

(412) 282-0864 Voice

1622 N. Main St., Butler, PA 16001
(412) 282-7965 FAX ■ (412) 282-2799 BBS

Trademark/Owner: OPTLINK/Windows/SLR Systems, Inc; Windows, MS-LINK, CodeView/Microsoft; Novell/Novell, Inc.; OS/2/IBM.

Circle 24 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

A coat of many colors

Create striking results with animation for the Edsun CEG/DAC

By Michael Abrash

In the beginning, there was CGA. Then there was EGA, then VGA, and finally SuperVGA. Today, a number of innovative enhancements to SuperVGA (SuperDuperVGA?) are reaching the market. It's too early to tell which, if any, of these technologies will become widely used, but one strong contender is Edsun's Continuous Edge Graphics Digital-to-Analog Converter (CEG/DAC).

The CEG/DAC is a plug-in replacement for any VGA's standard DAC (the part that converts pixel attributes from the VGA chip into the analog voltages representing colors that are sent to the monitor). Several manufacturers are currently shipping CEG/DAC-equipped boards; I expect to see the CEG/DAC widely used on SuperVGAs in the $200–$300 range, with resolutions up to 1,024 × 768 in 256 colors.

The CEG/DAC is purely a color enhancer; it does nothing to increase screen resolution. However, by making more colors available and thereby making it possible to anti-alias and smooth images, the CEG/DAC can increase the screen's *perceived* resolution. This is not automatic—the programmer must take advantage of CEG/DAC's expanded color set—but it is possible.

How many colors does the CEG/DAC make available? Theoretically, it is possible to get more than 100,000 colors onto a single 1,024 × 768, 256-color screen, but because the CEG/DAC expands the color set via commands embedded in the bitmap, the number of colors available in a single frame depends greatly on the nature of the displayed image. This will become clearer when we discuss pixel weighting and dynamic palette loading, the two mechanisms by which the CEG/DAC expands the color set. For now, the key point is that the CEG/DAC's expanded color set means that selecting the color of a pixel is no longer as simple as putting a single value in display memory (0 for black, 1 for blue, 2 for green, and so on). With the CEG/DAC, color selection becomes a sequence over time, and the color of a given pixel can depend on values stored earlier in the bitmap. This, alas, leads to complexity.

I think highly of the potential of the CEG/DAC, but I have worried that it will be particularly difficult to use for animation and other real-time applications. Having previously expressed those concerns in print, I thought it only fair that I actually write a CEG/DAC animation application.

My application uses all the major features of the CEG/DAC in the course of color-animating a shifting background and animating four bouncing balls. Each ball is a different color, and each has a specular highlight. More than 200 colors are left free for other purposes. As you will see, the CEG/DAC requires complex programming and has its limitations. The results, however, can be striking—and, in the final tally, results are what really matter.

Michael Abrash works for Metagraphics, of Scotts Valley, California. He is the author of Power Graphics Programming *and* Zen of Assembly Language. *Contact Michael at 7 Adirondack Street, South Burlington, VT 05403 or via MCI Mail, user name MABRASH.*

A QUICK PRIMER

The CEG/DAC has three modes of operation. Advanced-4, used in 16-color modes, makes available eight primary colors and eight weightings. Advanced-4 is your only choice in 16-color modes, but it is especially difficult to program. Basic-8, used in 256-color modes, makes available sixteen primary colors and eight weightings between pairs of colors. The simplest CEG/DAC mode, Basic-8 is good for line-oriented drawing.

Advanced-8, also used in 256-color modes, is CEG/DAC's most powerful mode. In this mode, 223 independently selectable colors are available at any one time. In addition, 32 weightings of color pairs are available, in steps of 1/31, from all one color to all the second color. Finally, any one DAC palette register can be reprogrammed on the fly wherever six pixels in a row of the same color occur in the bitmap. (For details on all three CEG/DAC modes, along with a broader overview, see the "Graphics Programming" column in the April and May 1991

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

issues of *Dr. Dobb's Journal*. Documentation and code for the CEG/DAC are available from Edsun in the *Edsun Continuous Edge Graphics Software Development Kit*. For now, these sources are your best bets for information; one can only hope that a book on programming the CEG/DAC appears soon.)

PALETTE LOADING

The special features of the Edsun chip operate by interpreting certain values in the pixel stream as opcodes rather than as color attributes. Information embedded in the bitmap can instruct the CEG/DAC to load any DAC register (palette entry) with a new color, thereby changing the color displayed by the corresponding pixel value. The sequence is shown in Figure 1. The first byte is the Edsun Dynamic Palette (EDP) opcode, which tells the CEG/DAC to interpret the next four pixels as RGB color components. This is followed by the number of the DAC register in which to store that color.

To have any effect, an EDP sequence must be scanned out as pixel information and sent to the CEG/DAC. An EDP sequence stored in off-screen memory would have no effect, because its bytes would never be sent to the CEG/DAC as pixels. Because EDP sequences are stored in the visible area of the bitmap, they are sent to the CEG/DAC *every frame*, so each EDP sequence loads a DAC register 60 or 70 times a second.

The instant the pixel containing the last byte of the EDP sequence arrives at the DAC, an EDP sequence changes the DAC register; thus, a register can be changed many times in a frame. For example, one EDP sequence can set DAC register 1 to green on scan line 0, and another can set DAC register 1 to blue on scan line 100; as a result, pixels with value 1 are green in the top half of the screen and blue in the bottom half. This procedure can increase the total color set available over an entire frame, as shown in Listings 1 and 2. (The number of colors simultaneously available is not increased, but clever allocation of objects and colors on the screen can work around this.)

What is on the screen while an EDP sequence is telling the CEG/DAC to load a DAC register? Whatever color happens to have been displayed for the pixel immediately preceding the EDP opcode. This means that EDP sequences can be placed in the bitmap only where there happen to be at least six pixels in a row of the same color, as is the case with the solid frame around the screen in Listing 1. Alternatively, EDP sequences could be stored in background areas of solid black or within solid areas of moving objects, or a black margin at each edge could be dedicated to EDP use, shrinking the usable screen area but increasing color selection.

I like EDP; it's my favorite feature of the CEG/DAC. EDP makes it possible to perform complex color animation (creating apparent motion by changing the colors displayed for a given pixel color) and, with proper planning, expands the number of independent colors available in the course of each frame. Both aspects of EDP are at work in Listing 1.

Note that the RGB values loaded by EDP are 8-bit values. In CEG mode, the CEG/DAC supports full 8-bit DAC registers, compared with the 6-bit DAC registers of a standard VGA. CEG mode can be enabled without enabling EDP. However, enabling EDP costs only one pixel value (191, the EDP opcode), and it's usually worth the loss of one of 224 simultaneous colors to be able to change DAC registers on the fly.

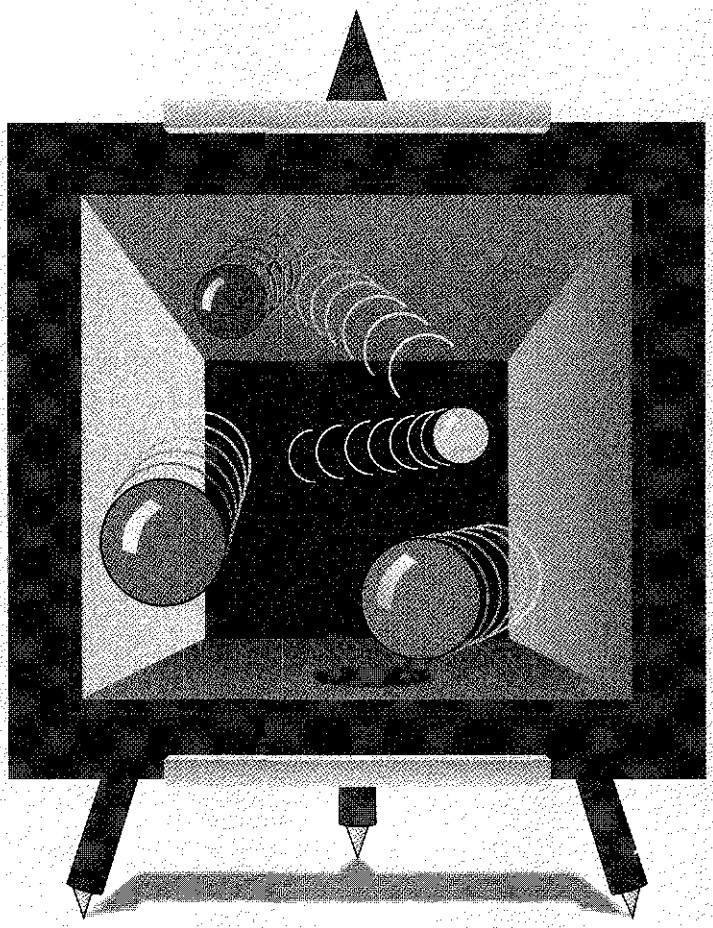


ILLUSTRATION BY ROB SCHUSTER

PIXEL WEIGHTING

Pixel weighting is the other primary feature of the CEG/DAC. A pixel weighting opcode specifies that a pixel's color is a weighted mix of the colors of two earlier pixels (see Figure 2). Table 1 shows the available mixes and shows that pixel weighting and EDP interpret 33 pixel values as opcodes rather than colors. This leaves 223 "normal" (non-EDP and non-pixel-weighted) colors available.

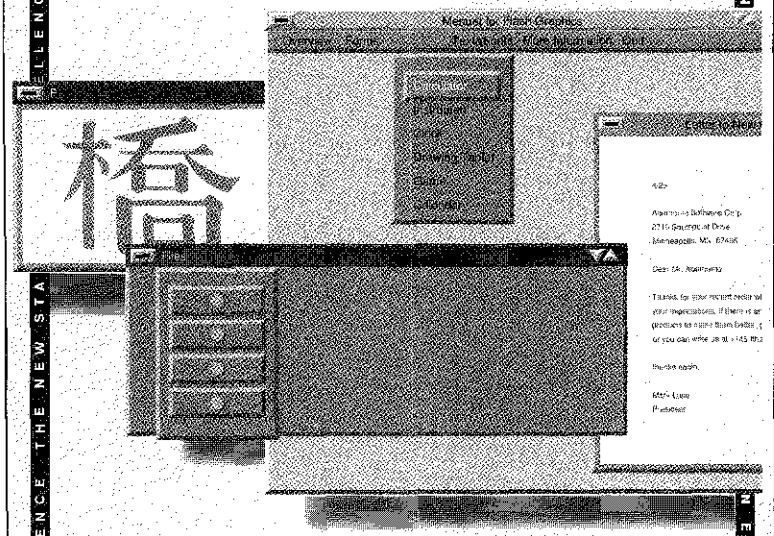
The CEG/DAC has a complex set of rules for interpreting various mixes of pixel colors and pixel weighting opcodes. The complications arise from Edsun's desire to be able to smooth narrow lines, which can cause considerable trouble when larger objects are pixel-weighted. For our purposes, we'll always start a pixel-weighting sequence with at least two pixels in a row of one color; this becomes the background color, color B. Then we'll have a single pixel of another color—the foreground color, color A; however, according to Edsun's rules, this pixel is still displayed in color B (don't ask why, that's just how it works). Following this will be at least two pixel-weighting opcodes. Therefore, at least three pixels are displayed in color B, followed by a minimum of two pixels displayed in mixes of colors A and B.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

THE NEW STANDARD IN EXCELLENCE THE NEW STANDARD IN EXCELLENCE

Menuet™

The New Standard in Excellence in GUI Design for DOS, 386 Protected Mode, and UNIX!



Versions for MetaWINDOW, GSS*CGI, SCO*CGI, and Zortech Flash Graphics

Includes Interface Design Tool, Menu Design Tool, a Font Editor, and an Icon Editor

Over 1,000 pages of printed documentation in 2 volumes, plus full on-line documentation

Easy to use - we've had customers create demos in 3 days, finished applications in 3 weeks

Simply outstanding aesthetics - Emulate WINDOWS, MOTIF, or your own unique look!

DOS Version: **$325** UNIX Version: **$650**

Call for your free demo!

303.494.8865

AUTUMN HILL SOFTWARE

Circle 25 on Reader Service Card

1145 ITHACA DRIVE, BOULDER, CO, 80303, FAX: (303) 494-7802 BBS: (303) 494-8868

FORMERLY ITHACA STREET SOFTWARE. ALL TRADEMARKS ARE THE PROPERTIES OF THEIR RESPECTIVE OWNERS.

THE NEW STANDARD IN EXCELLENCE THE NEW STANDARD IN EXCELLENCE

LISTING 1

```
/* Performs 256-color animation on the Edsun DAC, taking advantage of
   both pixel weighting (selecting pixel colors as weighted averages
   of the colors of nearby pixels) and EDP (the ability to embed
   information in the bitmap that reprograms the DAC's palette during
   the course of each frame). Requires a VGA equipped with an Edsun
   CEG/DAC. All C code compiled with Borland C++ 2.0, operating in C
   mode, and using the small model */

#include <conio.h>
#include <dos.h>
#include <stdio.h>
#include <math.h>

#define PI  3.141592
#define ADVANCED_8_EDP  15 /* CEG/DAC mode # for Advanced-8 mode with
                              EDP enabled */
#define FRAME_WIDTH 10     /* width of frame around bouncing area */
/* Bounds within which balls bounce */
#define TOP_EDGE  FRAME_WIDTH
#define BOTTOM_EDGE (200-FRAME_WIDTH-1)
#define LEFT_EDGE FRAME_WIDTH
#define RIGHT_EDGE (320-FRAME_WIDTH-1)

typedef struct _RGB {   /* defines RGB triplet for a color */
   unsigned char Red;
   unsigned char Green;
   unsigned char Blue;
} RGB;

typedef struct _Image {     /* defines a drawable image */
   int Width, Height;       /* dimensions */
   unsigned char *PixMap;   /* image bytes */
} Image;

typedef struct _Ball {      /* defines a bouncing ball */
   int X, Y;                /* location */
   int XDir, YDir;          /* direction of movement */
   Image *BallImage;        /* image to draw for ball */
} Ball;

int main(void);
int EnableCEGMode(int);
void DrawFrame(int,int);
void SetEDPAtRight(int,RGB*,int,int);
void FillRect(int,int,int,int,int);
void DisableCEGMode(void);
void WaitVerticalSync(void);
void DrawImage(int,int,Image *);
void SetEDPAtLeft(int,int,int,int,int);
void AdvanceAndDrawBalls(Ball**,int);
void Delay(void);
void SetCyclingAttributes(RGB*);
void InitializePalette(void);

/* Byte pattern for balls that bounce horizontally */
unsigned char HBallPixMap[] = {
     1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
     1,  1,  1,  1,  1,  2,  2,  2,  2,  1,  1,  1,  1,  1,
     1,  1,  1,  2,  2,  3,212,208,204,200,218,  1,  1,  1,
     1,  1,  2,  2,  3,214,210,205,200,198,200,220,  1,  1,
     1,  1,  2,  2,  3,216,212,207,202,200,204,218,  1,  1,
     1,  2,  2,  2,  3,218,214,208,206,205,210,216,  2,  1,
     1,  2,  2,  2,  3,220,216,214,212,210,216,220,  2,  1,
     1,  2,  2,  2,  3,222,220,218,216,216,218,222,  2,  1,
     1,  2,  2,  2,  2,  2,  3,221,220,221,  2,  2,  2,  1,
     1,  1,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  1,  1,
     1,  1,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  1,  1,
     1,  1,  1,  2,  2,  2,  2,  2,  2,  2,  2,  1,  1,  1,
     1,  1,  1,  1,  1,  2,  2,  2,  2,  1,  1,  1,  1,  1,
     1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
};
/* Byte pattern for ball that bounces diagonally */
unsigned char DBallPixMap[] = {
     1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
     1,  1,  1,  1,  1,  4,  4,  4,  4,  1,  1,  1,  1,  1,
     1,  1,  1,  4,  4,  5,212,208,204,200,218,  1,  1,  1,
     1,  1,  4,  4,  5,214,210,205,200,198,200,220,  1,  1,
     1,  1,  4,  4,  5,216,212,207,202,200,204,218,  1,  1,
     1,  4,  4,  4,  5,218,214,208,206,205,210,216,  4,  1,
     1,  4,  4,  4,  5,220,216,214,212,210,216,220,  4,  1,
     1,  4,  4,  4,  5,222,220,218,216,216,218,222,  4,  1,
     1,  4,  4,  4,  4,  4,  5,221,220,221,  4,  4,  4,  1,
     1,  1,  4,  4,  4,  4,  4,  4,  4,  4,  4,  4,  1,  1,
     1,  1,  4,  4,  4,  4,  4,  4,  4,  4,  4,  4,  1,  1,
     1,  1,  1,  4,  4,  4,  4,  4,  4,  4,  4,  1,  1,  1,
     1,  1,  1,  1,  1,  4,  4,  4,  4,  1,  1,  1,  1,  1,
```

continued

PRODUCED 2004 BY UNZ.ORG
REPRODUCTION PROHIBITED

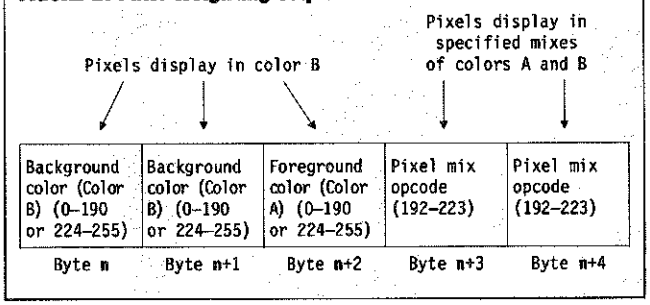
TABLE 1: Pixel values in Advanced-8 CEG mode 15

Pixel Value	Description
0	Attribute 0 (standard VGA; display the value in palette location 0)
1	Attribute 1 (standard VGA)
2	Attribute 2 (standard VGA)
.	
189	Attribute 189 (standard VGA)
190	Attribute 190 (standard VGA)
191	EDP (dynamic palette load) opcode
Pixel Weighting Opcodes	
192	100% (31/31) color A, 0% (0/31) color B
193	97% (30/31) color A, 3% (1/31) color B
194	94% (29/31) color A, 6% (2/31) color B
.	
222	3% (1/31) color A, 97% (30/31) color B
223	0% (0/31) color A, 100% (31/31) color B
224	Attribute 224 (standard VGA)
225	Attribute 225 (standard VGA)
.	
254	Attribute 254 (standard VGA)
255	Attribute 255 (standard VGA)

FIGURE 1: EDP sequence

191 (EDP opcode)	New red color component (0–255)	New green color component (0–255)	New blue color component (0–255)	Number of DAC register to load (0–190 or 224–255)
Byte n	Byte n+1	Byte n+2	Byte n+3	Byte n+4

FIGURE 2: Pixel-weighting sequence


Pixels display in color B
Pixels display in specified mixes of colors A and B
Background color (Color B) (0–190 or 224–255)
Background color (Color B) (0–190 or 224–255)
Foreground color (Color A) (0–190 or 224–255)
Pixel mix opcode (192–223)
Pixel mix opcode (192–223)
Byte n
Byte n+1
Byte n+2
Byte n+3
Byte n+4

If these conditions are observed (as in Listing 1), objects will draw predictably no matter what colors surround them, and the pixel-weighting opcodes will operate as specified in Table 1. If the conditions are *not* observed, surrounding colors may affect the object's appearance, or special line-drawing rules may come into play. If you don't want to observe the above limitations, refer to the CEG SDK documentation for more information.

I am not fond of pixel weighting—it's finicky and hard to program. It also creates serious problems when a pixel-weighted image is clipped or partially overwritten, because an interrupted pixel-weighting sequence often does not produce the same colors as the entire pixel-weighting sequence did. Nonetheless, pixel weighting does increase the available color set and can be useful for smoothing or for color gradients. Listing 1 illustrates the use of pixel weighting—and one of its drawbacks, as well.

FOLLOW THE BOUNCING BALLS

Listings 1 and 2 show a CEG/DAC animation application in which four balls of different colors, each smoothly highlighted, bounce around inside a multicolored frame against a background of smoothly scrolling colors. If you have access to a CEG/DAC, I suggest you run the application to give yourself a framework for the following discussion.

This example is not intended to be especially attractive, nor is it an ideal animation implementation. The balls are animated by building a blank fringe into each image so that when an image is moved, it overwrites completely and thereby erases itself at its last position. Overlapping images causes problems because the blank fringe is visible against the rearward image. Problems also can occur if an EDP sequence is scanned for video data while it's in the middle of being written to display memory (more about this later). For these and other reasons, it would be far better to perform page-flipped animation and to use a higher resolution than the 320 × 200 resolution of Listing 1. However, higher resolutions and page flipping would require more-complex code, and higher 256-color resolution would also require code that is specific to various VGA manufacturers. This article is bursting at the seams as it is, and the focus here is the CEG/DAC, so I've chosen to use the simplest 256-color mode and animation technique. Be aware, however, that techniques superior to those used in the sample program do exist.

That said, the sample application does illustrate a key tenet of CEG/DAC programming: tailor the application to the CEG/DAC's capabilities. Figure 3 (p. 35) shows how the sample application allocates the screen to take advantage of the CEG/DAC.

First, the 223 "normal" pixel attributes (which the DAC translates into colors) that are left when CEG mode is enabled are split up into several groups. Pixel attribute 0 is left as black; this is necessary to keep the overscan (screen border) area from becoming visible. Attribute 1 is the background color. Attributes 2 and 3 are used to draw the three horizontally bouncing balls, and 4 and 5 are used to draw the diagonally bouncing ball. Attributes 6–190 and 224–255 are set to a smoothly changing color sequence and then are used to draw a multicolored frame around the edge of the screen.

This frame is one way in which the application is tailored to the CEG/DAC; because it's more than five pixels wide and because the frame color doesn't change within a given scan line, you can store two EDP sequences on each scan line—one on each side of the frame—without altering the appearance of the frame. If the frame were 11 pixels wide, you could perform four EDP loads per scan line. By designing your applications to provide such areas of solid color, you can mask the loss of working screen space that EDP requires.

You could, of course, use attributes 6–190 and 224–255 for more than a mere frame. If a more sophisticated animation technique were used, you could use them to draw the background. You could also use them to draw non-CEG images or CEG images such as those drawn with attributes 2–5. Actually, you could accomplish

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
    1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
};

Image HBall = {14,14,HBallPixMap};  /* image for horz bouncing balls */
Image DBall = {14,14,DBallPixMap};  /* image for diag bouncing ball */
Ball Ball1 = {60, 30, -1, 0, &HBall}; /* the four balls to bounce */
Ball Ball2 = {150, 90, 1, 0, &HBall};
Ball Ball3 = {220, 150, -1, 0, &HBall};
Ball Ball4 = {100, 120, -1, -1, &DBall};
Ball* BallArray[] = {&Ball1, &Ball2, &Ball3, &Ball4};
#define NUM_BALLS (sizeof(BallArray)/sizeof(Ball*))
int main() {
   int i, GreenTemp, BlueTemp, AttributeStart, FrameCount;
   int FrameDelay, Temp;
   union REGS regset;
   RGB Attribute1Settings[384];  /* used to store the EDP settings for
                                    attribute 1 */
   /* Set to the standard 256-color VGA mode, mode 0x13, 320x200 */
   regset.x.ax = 0x0013; int86(0x10, &regset, &regset);

   /* Now enable Advanced-8 CEG mode, so pixel value 191 becomes EDP
      opcode, and pixel values 192-223 become pixel weighting opcodes */
   if (!EnableCEGMode(ADVANCED_8_EDP)) {
      /* This isn't a CEG/DAC; back to text mode and we're done */
      regset.x.ax = 0x0003; int86(0x10, &regset, &regset);
      fprintf(stderr, "No CEG/DAC installed\n");
      return(1);
   }

   InitializePalette();    /* set up the palette */

   /* Draw a frame, using the colors 6-190 & 224-255 that we just set
      up to produce a smooth color gradient from top to bottom */
   DrawFrame(6,FRAME_WIDTH);

   /* Fill inside the frame with attribute 1, which we'll use EDP to
      change from line to line */
   FillRect(LEFT_EDGE, TOP_EDGE, RIGHT_EDGE+1, BOTTOM_EDGE+1, 1);

   /* Set the array of values for EDP loads of attribute 1 and put the
      EDP sequences for the first 200 of those colors at the right
      edge of display memory, in the frame */
   SetCyclingAttributes(Attribute1Settings);
   SetEDPAtRight(1, Attribute1Settings, AttributeStart = 0,
         sizeof(Attribute1Settings)/sizeof(RGB));

   /* Set the EDP sequences for the colors of the three horz balls */
   SetEDPAtLeft(2, 0, 0, 0x20, 29);  /* top ball is blue */
   SetEDPAtLeft(3, 0, 0, 0xAF, 30);  /* top ball highlight color */
   SetEDPAtLeft(2, 0, 0x20, 0, 89);  /* middle ball is green */
   SetEDPAtLeft(3, 0, 0x9F, 0, 90);  /* middle ball highlight color */
   SetEDPAtLeft(2, 0x20, 0, 0, 149); /* bottom ball is red */
   SetEDPAtLeft(3, 0xCF, 0, 0, 150); /* bottom ball highlight color */

   /* Now cycle through a frame at a time, moving the balls and
      advancing the EDP settings for the background (attribute 1) so
      the background appears to move. Stop when a key is pressed */
   for (FrameCount = FrameDelay = 1; ; ) {   /* draw on first frame */
      WaitVerticalSync();  /* wait until the end of current frame */
      if (--FrameCount == 0) {
         FrameCount = FrameDelay;
         AdvanceAndDrawBalls(BallArray, NUM_BALLS);

         /* Change the attribute 1 (background) EDP loads by one
            position in Attribute1Settings, wrapping from start back
            to end. This has the effect of shifting the EDP load of
            attribute 1 with any given color by one scan line, so the
            whole background appears to move one scan line */
         if (AttributeStart == 0)
            AttributeStart = sizeof(Attribute1Settings)/sizeof(RGB)-1;
         else
            AttributeStart--;
         SetEDPAtRight(1, Attribute1Settings, AttributeStart,
               sizeof(Attribute1Settings)/sizeof(RGB));
                        /* set the new attribute 1 EDP sequences */
      }
      if (kbhit()) {
         Temp = getch();
         if ((Temp == 'f') || (Temp == 'F')) {
            if ((FrameDelay -= 5) <= 0) FrameDelay = 1; /* faster */
         } else if ((Temp == 's') || (Temp == 'S')) {
            FrameDelay += 5;  /* slower */
         } else {
            break;   /* done */
         }
      }
   }

   DisableCEGMode();
   regset.x.ax = 0x0003; int86(0x10,&regset,&regset);  /* text mode */
return(0);
}

/* Enables the desired CEG/DAC mode. Returns 1 for success, 0 if no
   CEG/DAC is installed. */
int EnableCEGMode(int mode) {
   WaitVerticalSync();
   /* Write the CEG enable sequence, which is "CEGEDSUN" followed by
      the mode byte, to palette location 222 */
   outp(0x3C7,222); Delay(); outp(0x3C9,'C'); Delay();
   outp(0x3C9,'E'); Delay(); outp(0x3C9,'G'); Delay();
   outp(0x3C7,222); Delay(); outp(0x3C9,'E'); Delay();
   outp(0x3C9,'D'); Delay(); outp(0x3C9,'S'); Delay();
   outp(0x3C7, 222); Delay(); outp(0x3C9,'U'); Delay();
   outp(0x3C9,'N'); Delay();
   outp(0x3C9, mode); Delay();   /* write the CEG mode */

   /* At this point, CEG mode should be enabled. Make sure this really
      is a CEG/DAC */
   outp(0x3C6, 0xFF); Delay();   /* enable all DAC mask bits */
   if ((inp(0x3C6) & 0x70) == 0x70)
      return(0);  /* no version # bit is 0; this is not a CEG/DAC */
   else
      return(1);  /* this is a CEG/DAC, and CEG mode is now enabled */
}

/* Disable CEG mode by writing to palette location 223 */
void DisableCEGMode() {
   WaitVerticalSync();
   outp(0x3C8, 223); outp(0x3C9, 0); outp(0x3C9, 0); outp(0x3C9, 0);
}

/* Wait for the leading edge of vertical sync */
void WaitVerticalSync() {
   while (inp(0x3DA) & 0x08) ;      /* wait for not vertical sync */
   while (!(inp(0x3DA) & 0x08)) ;   /* wait for vertical sync */
}

/* Advance each ball and draw it (the blank fringe around each ball
   erases it at the old location) */
void AdvanceAndDrawBalls(Ball** BallPtrPtr, int NumBalls) {
   int i, Temp;
   Ball *BallPtr;

   for (i=0; i<NumBalls; i++) {
      BallPtr = *BallPtrPtr++;   /* point to current ball */
      /* Advance the X coordinate, bouncing if frame reached */
      Temp = BallPtr->X + BallPtr->XDir;
      if ((Temp < LEFT_EDGE) ||
            ((Temp + BallPtr->BallImage->Width) > RIGHT_EDGE)) {
         BallPtr->XDir = -BallPtr->XDir;  /* bounce in X dir */
         Temp += BallPtr->XDir;
      }
      BallPtr->X = Temp;
      /* Advance the Y coordinate, bouncing if frame reached */
      Temp = BallPtr->Y + BallPtr->YDir;
      if ((Temp < TOP_EDGE) ||
            ((Temp + BallPtr->BallImage->Height) > BOTTOM_EDGE)) {
         BallPtr->YDir = -BallPtr->YDir;  /* bounce in Y dir */
         Temp += BallPtr->YDir;
      }
      BallPtr->Y = Temp;
      /* Draw the ball at the new location */
      DrawImage(BallPtr->X, BallPtr->Y, BallPtr->BallImage);
   }
}

/* Used to provide a brief delay between OUTs */
void Delay() {
}

/* Set AttributeArray to contain a color sequence gradually
   cycling through each primary color */
void SetCyclingAttributes(RGB* AttributeArray) {
   int i;

   for (i=0; i<64; i++) {
      AttributeArray[i].Red = i;      /* ascending red */
      AttributeArray[i].Green = 0;
      AttributeArray[i].Blue = 0;
      AttributeArray[i+64].Red = 63-i; /* descending red */
      AttributeArray[i+64].Green = 0;
      AttributeArray[i+64].Blue = 0;
      AttributeArray[i+128].Red = 0;  /* ascending green */
      AttributeArray[i+128].Green = i;
      AttributeArray[i+128].Blue = 0;
      AttributeArray[i+192].Red = 0;  /* descending green */
      AttributeArray[i+192].Green = 63-i;
      AttributeArray[i+192].Blue = 0;
      AttributeArray[i+256].Red = 0;  /* ascending blue */
      AttributeArray[i+256].Green = 0;
      AttributeArray[i+256].Blue = i;
```

continued

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

the multicolored effect of the frame equally well by drawing it in just one attribute and performing an EDP load on each line, as is done with attribute 1 (which we'll get to next). I've drawn the frame with hundreds of attributes to show that they are indeed available. When you watch the sample application in action, remember that only five palette entries are used to draw the entire background and the balls; the other 218 attributes, including all the colors in the frame, are available for any CEG-style use you desire.

COLOR ANIMATION WITH EDP

The sample application sets the background to a smooth color gradient by filling the interior of the frame with attribute 1, then storing in the right edge of the frame EDP sequences that change the setting of DAC register 1 at the end of each scan line. The colors of the background pixels are then made to move over time by shifting by one the scan line on which each EDP sequence resides; this changes the color interpretation of the background pixels' attribute of 1, although the values of the background pixels' bytes in display memory do not change. The net effect is one of a colored background moving past in a downward direction. This can be used to create the illusion that the foreground objects, along with the viewing window, are moving upward.

Again, the application is designed to allow EDP to perform color animation; a vertically striped background wouldn't work at all. EDP is also used to affect the colors of the pixel-weighted foreground objects, as described below.

PIXEL WEIGHTING IN ACTION

As described above, each ball that moves in the foreground is handled via the simple animation technique of building into the ball image a blank fringe that overwrites the image at its previous position. In keeping with the rules specified above for pixel-weighting sequences, each ball has at least two pixels of attribute 2 (the ball attribute) at the left edge of each scan line, followed by one pixel of attribute 3 (the highlight attribute), followed by at least two pixels of pixel-weighting mixes.

The initialization of **HBallPixMap** in Listing 1 contains

- the blank fringe (composed of attribute 1, the background color);
- the ball attributes (attribute 2, always at least two pixels wide at the left);
- the highlight attributes (attribute 3, always one pixel wide immediately to the right of attribute 2; this pixel actually displays on the screen as attribute 2); and
- the pixel-weighting opcodes, in the range 192–223, that draw the highlighted area.

Within the pixel-weighted area, lower values weight pixels more toward attribute 3, and hence make them more highlighted. Only two attributes are used to draw each ball, but more than a dozen colors are produced, thanks to pixel weighting.

There are four balls, drawn in four

PROFESSIONAL SERIAL COMMUNICATION MADE EASY

DON'T BE FOOLED BY THE PRICE

COMM-DRV is a professional serial communication library that comprises a complete serial I/O library, a true MSDOS RS232 device driver, an RS232 TSR, a serial port monitor, and a dumb terminal emulator. It is second to none!!!

VERY EASY TO USE !

- 25+ serial ports sharing one or more IRQs. Remap COM ports to any address.
- Supports speeds up to 115.2 kilobaud.
- Hardware/Software Handshake - RTS/CTS DSR/DTR XON/XOFF
- Installable as a device driver or as a TSR (can be de-installed from RAM).
- Can be linked into application or used via TSR or device driver.
- Supports adjustable communication buffer sizes on every port.
- Runs under Desqview, Omniview, VPIX, Windows 3.0, and most multitaskers.
- Interface to DOS via INT21, INT14, C's open(), read(), write(), close(), ioctl().
- Selectively interrupt user routine on reception, transmission, & status change.
- Supports most multiport cards including Digiboards & Intel Hub card.
- Coexistence of different multiport, singleport, & smart cards.
- Supports the FOSSIL interface used on most bulletin boards (e.g. OPUS)
- Supports all file transfer protocols.
- 8250, 16450, 16550 (including FIFO buffers), etc. support.
- Compatible with all languages (C, QBASIC, PASCAL, COBOL, etc.)
- Customization provided upon request.

Driver, Library, Sources $89.95
Driver, Library, Sources, & 4 PORT SERIAL CARD $199.95
Windows 3.0 Driver For Multiport Cards $99.95

TEL.(713)498-4832 FAX (713)-568-3334
1-800-966-4832 BBS (713)-568-6401
VISA/MC/AMEX/PO/COD/CHECK 24 Hours/Day 7 Days/Week
Willies' Computer Software Co.
2470 S. Dairy Ashford, Suite 188 Houston, Tx. 77077

Circle 26 on Reader Service Card

ProGraphx Toolbox™

New Version 2.0
Write for a Free Demo Disk

NEW FEATURES INCLUDE:
•320x200 EGA mode for games
•Icon Animation Editor
•Great for games, real-time,
•or detail-oriented graphics

Over 50 highspeed graphics routines for EGA/VGA 16 color modes, including proprietary color icon routines. **ProGraphx Image Editor** is a complete color icon paint package which allows you to edit and view up to 50 icons at once. Simply paint an icon, then use the **ProGraphx Toolbox™** routines to display it with amazing speed. Over 500 sample icons for use in your own programs. Includes **Operation: Airlift**, a full-screen scrolling arcade style game with source code. Works with ANY C or Turbo Pascal.
All routines written in assembly forfastest performance

SPECIAL OFFER
$99 with source
No Royalties!

To order, write:
Peder Jungck
1019 West Lovers Lane
Arlington, TX 76013
Include $5 S&H

Circle 27 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

colors, but there are actually only two ball images, using a total of four attributes. EDP is used to change the colors associated with the attributes (2 and 3) of each of the three horizontally bouncing balls, so the same image can be used for each. Of course, this works only because the balls never share the same scan line. The diagonally bouncing ball, which does share scan lines with the horizontal balls, must use two different attributes; if it used the same attributes, it would change colors as it moved up and down the screen past the various EDP loads. Again, the application is designed so that a single ball image can be used to draw three different balls in three different colors.

THE DARK SIDE

I mentioned the potential problems that occur when a pixel-weighting sequence is interrupted. This phenomenon is present in the sample application when balls cross, but it isn't noticeable because the balls are small and moving fast and because relatively low-intensity colors are used. To see the problem more clearly, you can press the *S* key several times to slow the animation down when two balls are about to intersect. (Press *F* to speed things back up.) As the balls slowly cross, you'll see two things happen: bright sparkles sometimes appear in the ball to the left, and at least part of the highlighted area in the ball to the right sometimes changes colors. (You'll also see the blank fringe around one ball, but that's an artifact of the animation approach, not the CEG/DAC.)

The sparkles to the left happen when the pixel-weighting mix is truncated to only one pixel-weighting opcode; this causes the special narrow-line mode to kick in, often producing too-bright or incorrectly colored pixels. The color change in the highlighted area occurs when the pixels that set the foreground and background colors for a pixel-weighting sequence in one ball are overwritten by the other ball; the colors of the other ball then serve as the mix sources to the pixel-weighting sequence, and the mixed pixels change color. If the effects aren't evident, change the EDP sequences to make all the highlighted colors bright white; the color problems will then be evident, even in normal operation.

In this case, I intentionally designed the application around a weakness of the CEG/DAC instead of a strength, to make the problem obvious. You could choose to ignore the problem, but given that high image quality is the whole point of the CEG/DAC, that seems self-defeating, especially for larger or slower-moving images. You can work around the problem by keeping the balls from intersecting—by bouncing them off each other, designing their paths so that they never intersect, removing balls from the screen when struck, and the like.

None of these solutions is ideal, however, which is why I don't much care for pixel weighting in animation; it creates as many problems as it solves. My preferred solution is to use EDP for drawing the balls; there's plenty of space for additional EDP sequences in the frame area, and it gives you superior color control with no intersection problems. The flip side is that drawing the balls with EDP

```
        AttributeArray[i+320].Red = 0;  /* descending blue */
        AttributeArray[i+320].Green = 0;
        AttributeArray[i+320].Blue = 63-i;
    }
}

/* Set up the palette with the colors for the outer frame (colors 6
   through 190 and 224 through 255) cycling through combinations of
   the three primary colors as determined by out-of-phase sine waves.
   (There's nothing profound about these colors; it just makes an
   attractive, multicolored border.) Also set attributes 4 and 5 to
   the colors for the ball that bounces diagonally, and sets attrs
   0-3 to black (attributes 1-3 will be altered via EDP later) */
void InitializePalette() {
    int i, ColorEntry;
    RGB PaletteTemp[256];  /* used to pass palette settings to BIOS */
    union REGS regset;
    struct SREGS sregset;

    for (i=0; i<4; i++) {
        PaletteTemp[i].Red = 0;     /* black */
        PaletteTemp[i].Green = 0;
        PaletteTemp[i].Blue = 0;
    }
    PaletteTemp[4].Red = 0;     /* the diagonal ball is cyan */
    PaletteTemp[4].Green = 0x20;
    PaletteTemp[4].Blue = 0x20;
    PaletteTemp[5].Red = 0;     /* highlight color for diagonal ball */
    PaletteTemp[5].Green = 0xAF;
    PaletteTemp[5].Blue = 0xAF;
    ColorEntry = 0;
    for (i=6; i<=255; ColorEntry++, i = ((i==190) ? 224 : i+1)) {
                                        /* skip i from 190->224 */
        PaletteTemp[i].Red = abs((int)(255.0 *
            sin((double)(ColorEntry+67)/217.0*PI)));    /* Red */
        PaletteTemp[i].Green = abs((int)(255.0 *
            sin((double)ColorEntry/217.0*PI)));         /* Green */
        PaletteTemp[i].Blue = abs((int)(255.0 *
            sin((double)(ColorEntry+133)/217.0*PI)));   /* Blue */
    }
    regset.x.ax = 0x1012;   /* set block of DAC registers function */
    regset.x.bx = 0;        /* set register 0 first */
    regset.x.cx = 190-0+1;  /* # of registers to set */
    regset.x.dx = (unsigned int)&PaletteTemp[0];
                            /* offset of array from which to set */
    sregset.es = _DS;       /* segment of array from which to set */
    int86x(0x10, &regset, &regset, &sregset); /* attributes 0->190 */
    regset.x.ax = 0x1012;       regset.x.bx = 224;
    regset.x.cx = 255-224+1;    sregset.es = _DS;
    regset.x.dx = (unsigned int)&PaletteTemp[224];
    int86x(0x10, &regset, &regset, &sregset); /* attributes 224->225 */
}
```

LISTING 2

```
; Assembly routines for Edsun CEG/DAC animation program.

SCREEN_SEGMENT  equ     0a000h
SCREEN_WIDTH    equ     320
SCREEN_HEIGHT   equ     200

; Draws a frame around the screen, drawing top scan line of the frame in
; color StartColor, the next in color StartColor+1, and so on up through
; color 190, then jumping to color 224 and continuing to color 255, and back
; around to color StartColor if the screen has that many scan lines.
;
; C near callable as: void DrawFrame(int StartColor, int FrameWidth);
;
; All assembly code tested with TASM 2.

parms   struc
        dw      2 dup (?)       ;pushed BP & return address
StartColor dw   ?               ;color of first scan line
FrameWidth dw   ?               ;width of frame
parms   ends

; Macro to advance to the next color, going from StartColor to 190,
; then to 224 up through 255, and then back to StartColor.
CYCLE_COLOR     macro
        local   NoTurnover,CycleColorDone

        inc     al              ;advance the color
        jnz     NoTurnover      ;didn't turn over from 255 to 0
        mov     al,byte ptr StartColor[bp] ;did turn over, force to
        jmp     CycleColorDone             ;  StartColor
NoTurnover:
        cmp     al,191          ;time to jump to second block?
        jnz     CycleColorDone  ;no, the color is fine as is
        mov     al,224          ;yes, jump to first color in 2nd block
CycleColorDone:
```

continued

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Windows 3.0 DLL
Now Included!
ROYALTY FREE

Objective dBASE programmers try C++

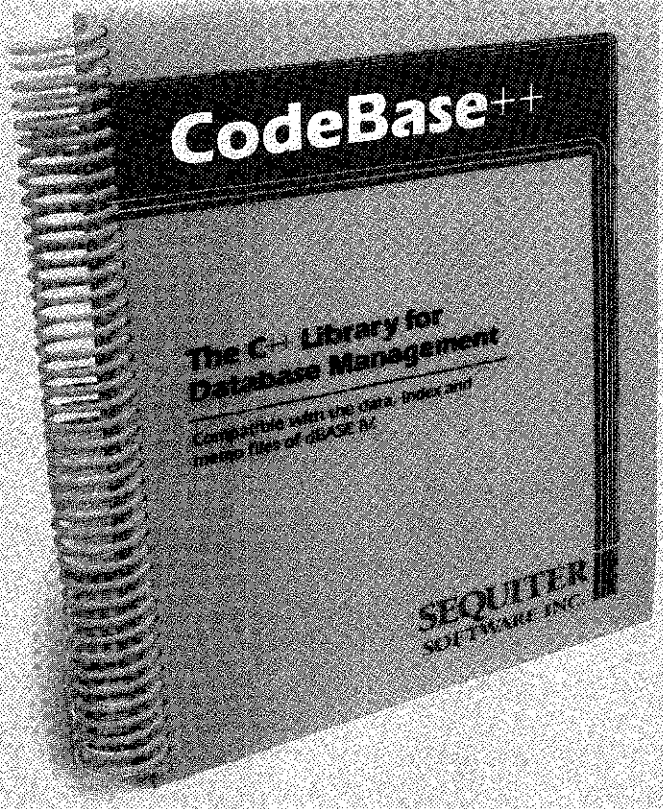
Use CodeBase++ with C++ to build multi-user, dBASE compatible programs. Take advantage of object orientated programming to quickly create maintainable, readable software.

dBASE Compatible

Use CodeBase++ classes to access and change the data, index and memo files of dBASE IV and III. Use other classes to manipulate strings, sort at high speed, perform date arithmetic, evaluate dBASE expressions, manipulate linked lists and manage memory.

Portable

Port your programs between DOS, Microsoft Windows, and OS/2. Keep all the profits as you distribute your programs royalty free.



Fast

Watch as your C++ programs execute much faster than corresponding dBASE, Clipper or FoxPro programs. Be surprised at the small executables and low memory requirements. Examine the C++ source code, which is completely included, to learn good object orientated programming techniques.

Order Today

Order today at $295 and discover why Sequiter Software Inc. and most dealers offer a 60-day money back guarantee.

SEQUITER SOFTWARE INC.

Call (403) 448-0313

Fax (403) 448-0315

P.O. Box 5659, Station L, Edmonton, Alberta, Canada T6C 4G1

Circle 16 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

requires the use of more palette entries than does pixel weighting.

IMPORTANT SAFETY TIP

As shown in Listing 2, EDP sequences are written to memory in a peculiar order: DAC register number first, then EDP opcode, and finally the RGB value to load. The reason for this is not obvious; to be honest, I discovered it only by chance.

Originally, I wrote EDP sequences to display memory in their natural order: EDP opcode first, then RGB value, and finally DAC register number, as shown in Figure 1. Usually, this worked fine. Occasionally, however, attribute 0 would change from black to dim red, and the overscan (border) region would become visible.

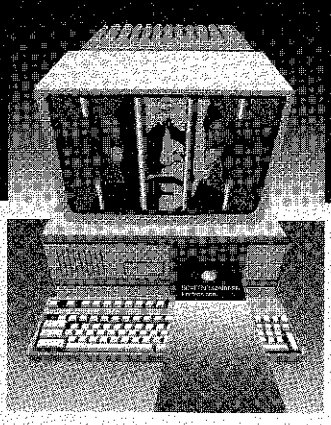
Here's what was happening. On occasion, I would write the EDP opcode and the RGB values to memory *just* before the VGA scanned those bytes of memory out and sent them to the DAC as pixel data. Therefore, the DAC would receive the EDP opcode and the RGB value intended for, say, DAC register 1. However, before the program got around to writing the value 1 (or whatever the intended DAC register was) at the end of the EDP sequence, the VGA would scan out that byte—which was still 0, because the number of the DAC register to load hadn't yet been written to it.

The CEG/DAC saw this sequence of pixels: EDP opcode, red value, green value, blue value, 0. The CEG/DAC dutifully did what that sequence specified, which was to load DAC register 0 (not 1) with the RGB value. By the next frame, of course, the EDP sequence was complete in memory, and the correct DAC register (register 1) was loaded. But by then the damage had been done; register 0, which I had not intended to modify, was changed.

The workaround (Listing 2) is to set the register number *first*, then the EDP opcode, and then the RGB values. That way, the EDP opcode can't help but be associated with the correct DAC register. The RGB values must be written after the EDP opcode, because if one of them is coincidentally the value 191 (the EDP opcode), and if the timing were exactly right, it could be incorrectly interpreted as the EDP opcode for a single frame. This sequence could cause an unintended DAC register to be loaded.

Another solution is to write EDP sequences only immediately following the vertical sync pulse, when no information is being sent to the DAC. However, it is hard to use this approach to write multiple EDP sequences reliably on computers of widely varying speeds. As is generally the case with the CEG/DAC, the best solution is to use page flipping, so intermediate bitmap states can't be misinterpreted. (Page flipping also avoids the possibility of partially written pixel-weighting sequences being sent to the CEG/DAC, which can result in one-frame "sparkles" as incorrect pixel values are generated.)

In sum, if there's any chance that the display memory to which you're writing



Free Yourself!
With Screen Manager Professional™

Concentrate on the substance of your programs with the advanced interface design library for C programmers.

Screen Manager Professional gives your applications the finest bullet proof windowing interface available. All the functions have been carefully designed and tested with the serious programmer in mind The functions are easy to use and remember. The elegant and powerful functions provide the fastest possible response and smallest possible size. Free yourself to concentrate on the substance of your program. Over 200 functions provide all the windowing tools needed to develop serious applications whether you are a professional or a hobbyist.

SCREEN MANAGER Professional™

MAGEE™

SMP gives your applications:

- Windowing
- Menus
- Mouse Support
- Field oriented data entry
- Interactive field building utility
- Keyboard Support Functions
- Context Sensitive Help with Hypertext
- Low RAM Overhead / High Speed
- User attachable validation functions

Linkable with: Microsoft C - Zortech C, C++ - Quick C
Turbo C, C++ - Watcom C

Only $349.95 **Source code available**

Call today for a free demonstration diskette
1-800-662-4330

P.O. Box 1587
Norcross, GA 30091 USA
404-446-6611
BBS 404-446-6650

Circle 29 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
        endm

        .model small
        .code
        public  _DrawFrame
_DrawFrame      proc    near
        push    bp              ;preserve caller's stack frame
        mov     bp,sp           ;point to local stack frame
        push    si              ;preserve caller's register variables
        push    di
        cld
        mov     ax,SCREEN_SEGMENT
        mov     es,ax
        sub     di,di           ;point to the first byte of the frame
        mov     al,byte ptr StartColor[bp] ;initial color
; First, draw the top of the frame.
        mov     dx,FrameWidth[bp] ;height of top part of frame
        mov     bx,SCREEN_WIDTH ;# of bytes per scan line
DrawTopLoop:
        mov     cx,bx
        rep     stosb           ;draw one scan line of the frame top
        CYCLE_COLOR             ;advance to the next color
        dec     dx              ;count down frame top scan lines
        jnz     DrawTopLoop
; Next, draw the sides of the frame.
        mov     cx,FrameWidth[bp]
        mov     si,cx           ;SI = FrameWidth for later
        add     cx,cx           ;FrameWidth * 2
        mov     dx,SCREEN_HEIGHT ;height of entire frame
        sub     dx,cx           ;height of sides =
                                ; SCREEN_HEIGHT - (FrameWidth * 2)
        sub     bx,cx           ;distance from one side to other =
                                ; SCREEN_WIDTH - (FrameWidth * 2)
DrawSidesLoop:
        mov     cx,si           ;width of each side = FrameWidth
        rep     stosb           ;draw one scan line of the left side
        add     di,bx           ;point to the start of the right side
        mov     cx,si           ;width of each side = FrameWidth
        rep     stosb           ;draw one scan line of the right side
        CYCLE_COLOR             ;advance to the next color
        dec     dx              ;count down frame top scan lines
        jnz     DrawSidesLoop
; Finally, draw the bottom of the frame.
        mov     bx,SCREEN_WIDTH ;# of bytes per scan line
DrawBottomLoop:
        mov     cx,bx
        rep     stosb           ;draw one scan line of frame bottom
        CYCLE_COLOR             ;advance to the next color
        dec     si              ;count down frame bottom scan lines
        jnz     DrawBottomLoop
        pop     di              ;restore caller's register variables
        pop     si
        pop     bp              ;restore caller's stack frame
        ret
_DrawFrame      endp

; Sets the 5 rightmost pixels on each scan line to perform an EDP load
; of the corresponding element from AttributeSettings to the DAC register
; specified by DACRegisterToLoad. (Every scan line reloads DACRegisterTo-
; Load.) Stores each EDP sequence in display memory in this order: DAC
; register #, the EDP opcode, and finally RGB color to load, in order to
; avoid accidentally loading the wrong DAC register if those bytes happen
; to be scanned just as the EDP sequence is half-written to memory.
;
; C near-callable as: SetEDPAtRight(int DACRegisterToLoad,
;       RGB* AttributeSettings, int FirstEntryToSet, int LastEntry);
SEARparms struc
        dw      2 dup (?)       ;pushed BP & return address
DACRegisterToLoad dw    ?       ;DAC register to perform EDP load to
AttributeSettings dw    ?       ;palette value table to load from
                                ; (array of RGB triplets)
FirstEntryToSet dw      ?       ;# of first entry in AttributeSettings
                                ; from which to load
LastEntry       dw      ?       ;# of last entry in AttributeSettings
SEARparms ends
        public _SetEDPAtRight
_SetEDPAtRight  proc    near
        push    bp              ;preserve caller's stack frame
        mov     bp,sp           ;point to local stack frame
        push    si              ;preserve caller's register variables
        push    di
        cld
        mov     ax,SCREEN_SEGMENT
        mov     es,ax
        mov     di,SCREEN_WIDTH-5 ;point to the first byte at which to
                                ; store EDP sequence on scan line 0
```

continued

Passive Backplane CPUs

PC Tech manufactures a full line of modular passive backplane high performance PC compatible CPUs.

Passive backplane based systems have many advantages, including ease of assembly, flexibility, fast service (by swapping boards), and simple upgrades to new or faster processors.

Contact us for more information, pricing, and benchmark results.

80386SX: 20 MHz, 512K through 16M DRAM

80386DX: 25 MHz & 33 MHz, 1M - 32M DRAM, 128K cache module optional

80486: 25 MHz & 33 MHz, 1M - 32M DRAM, 256K cache module optional

Call, write, or FAX for complete specifications and retail or OEM terms. All PC Tech products are available in complete systems as well as components for retail, OEM, and private label sales.

Designed, Manufactured, Sold and Serviced by:

PC Tech Inc.

Voice . . . (612) 345-4555
FAX . . . . (612) 345-5514
Modem . . (612) 345-4656

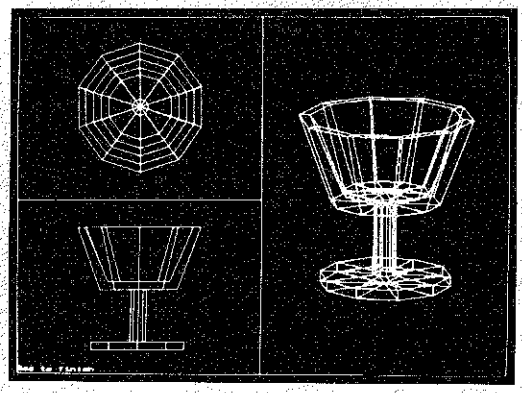
907 N. 6th St., Lake City, MN 55041

Circle 30 on Reader Service Card

THE ULTIMATE CAD/CAM ENGINE

TurboGeometry Library 3.0!

Helps You build it right THE FIRST TIME!



Over 300 2D & 3D device independent geometric routines. Surfacing; Spherical & Solid Geometry; Tangents; Curves; Hidden Line; Polygons; Perspectives; 2D & 3D Clipping; Link List Library & more. In C or Pascal, w/source. 400 pg manual. $199.95, Foreign $225.00. Prices incl S&H. 30 day guarantee. VISA, MC, PO, Chk.

2116 E. Arapaho Rd.
Suite 487
Richardson, TX 75081
USA

Disk Software, Inc.

FAX or Phone
214-423-7288
Toll Free
1-800-635-7760

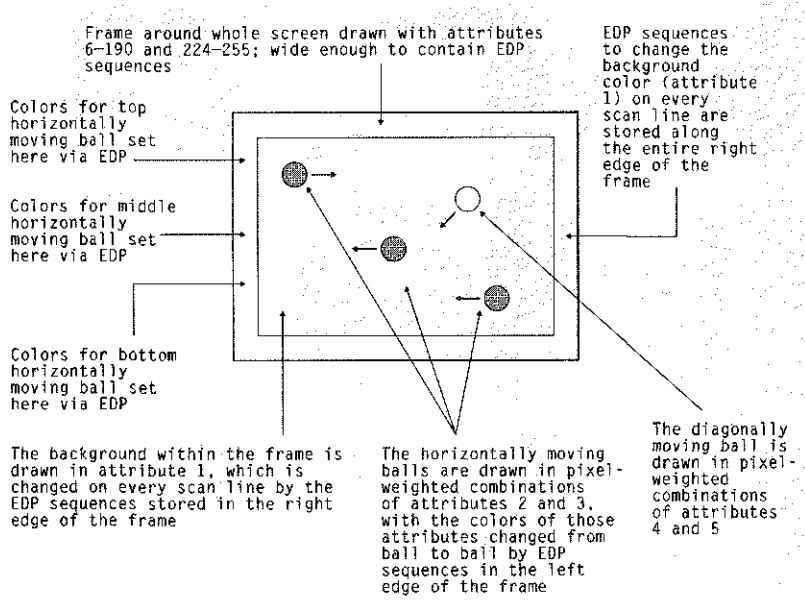
Circle 31 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
        mov     si,FirstEntryToSet[bp]
        add     si,si           ;first entry index * 2
        add     si,FirstEntryToSet[bp] ;first entry index * 3
        mov     cx,AttributeSettings[bp]
        add     si,cx           ;point to first entry to load
        mov     bx,LastEntry[bp]
        add     bx,bx           ;last entry index * 2
        add     bx,LastEntry[bp] ;last entry index * 3
        add     bx,cx           ;point to byte after last entry to load
        mov     ah,byte ptr DACRegisterToLoad[bp] ;DAC reg to load
        mov     cx,SCREEN_HEIGHT ;put one EDP sequence on each line
SetEDPLoop:
        mov     es:[di+4],ah    ;set the # of the DAC register to load
        mov     al,191          ;EDP opcode
        stosb                   ;put EDP opcode in bitmap
        movsb                   ;copy over the RGB value to load with
        movsb                   ; this EDP opcode
        movsb
        add     di,SCREEN_WIDTH-4 ;point to the first byte of the EDP
                                ; sequence on the next scan line
        cmp     si,bx           ;have we wrapped off the end of the
                                ; attribute array?
        jb      EDPLoopBottom   ;no
        mov     si,AttributeSettings[bp] ;point to start of attr array
EDPLoopBottom:
        loop    SetEDPLoop
        pop     di              ;restore caller's register variable.
        pop     si
        pop     bp              ;restore caller's stack frame
        ret
_SetEDPAtRight  endp

; Fills rectangle from (StartX,StartY) to (EndX-1,EndY-1); row at EndY and
; column at EndX are not drawn. No clipping or error checking is performed.
;
; C near-callable as: FillRect(int StartX, int StartY, int EndX,
;       int EndY, int FillColor);

FRparms struc
        dw      2 dup (?)       ;pushed BP & return address
StartX  dw      ?               ;left edge of fill rect
StartY  dw      ?               ;top edge of fill rect
EndX    dw      ?               ;right edge of fill rect + 1
EndY    dw      ?               ;bottom edge of fill rect + 1
FillColor dw    ?               ;color with which to fill rect
FRparms ends

        public _FillRect
_FillRect       proc    near
        push    bp              ;preserve caller's stack frame
        mov     bp,sp           ;point to local stack frame
        push    si              ;preserve caller's register variables
        push    di
        cld
        mov     ax,SCREEN_SEGMENT
        mov     es,ax
        mov     bx,SCREEN_WIDTH
        mov     ax,StartY[bp]
        mov     si,EndY[bp]
        sub     si,ax           ;# of scan lines to fill.
        mul     bx              ;start of top scan line of rect
        mov     dx,EndX[bp]
        mov     di,StartX[bp]
        sub     dx,di           ;# of bytes across rect
        add     di,ax           ;ES:DI = upper left byte of rect
        sub     bx,dx           ;distance from end of one rect scan
                                ; line to start of next
        mov     al,byte ptr FillColor[bp] ;color with which to fill
FillLoop:
        mov     cx,dx           ;# of bytes across
        rep     stosb           ;fill one scan line
        add     di,bx           ;point to start of next line to fill
        dec     si              ;count off scan lines
        jnz     FillLoop
        pop     di              ;restore caller's register variables
        pop     si
        pop     bp              ;restore caller's stack frame
        ret
_FillRect       endp

; Copies the image pointed to by ImagePtr with the upper left corner
; at (StartX,StartY). No clipping or error checking is performed.
;
; C near-callable as: DrawImage(int StartX, int StartY, Image* ImagePtr);

DIparms struc
        dw      2 dup (?)       ;pushed BP & return address
DIStartX dw     ?               ;left edge of block destination
DIStartY dw     ?               ;top edge of block destination
ImagePtr dw     ?               ;pointer to Image struc to draw
DIparms ends

Image   struc
Width   dw      ?
Height  dw      ?
PixMap  dw      ?
Image   ends

        public _DrawImage
_DrawImage      proc    near
        push    bp              ;preserve caller's stack frame
        mov     bp,sp           ;point to local stack frame
        push    si              ;preserve caller's register variables
        push    di
        cld
        mov     ax,SCREEN_SEGMENT
        mov     es,ax
        mov     ax,SCREEN_WIDTH
        mul     DIStartY[bp]    ;start of top scan line of rect
        add     ax,DIStartX[bp]
        mov     di,ax           ;ES:DI = upper left byte of rect
        mov     si,ImagePtr[bp] ;point to Image struc to draw
        mov     dx,[si].Width   ;# of bytes across rect to draw
        mov     ax,[si].Height  ;# of scan lines to copy
        mov     si,[si].PixMap  ;image bytes to copy
        mov     bx,SCREEN_WIDTH
        sub     bx,dx           ;distance from end of one rect scan
                                ; line to start of next
CopyLoop:
        mov     cx,dx           ;# of bytes across
        rep     movsb           ;copy one scan line
        add     di,bx           ;point to next line to copy to
        dec     ax              ;count off scan lines
        jnz     CopyLoop
        pop     di              ;restore caller's register variables
        pop     si
        pop     bp              ;restore caller's stack frame
        ret
_DrawImage      endp

; Puts the EDP sequence that loads the specified RGB color into the
; specified DAC register (that is, palette location, changing the
; color of the corresponding attribute) in the second through sixth
; pixels of the specified scan lines. Like SetEDPAtRight, stores EDP
; sequences in DAC register #/EDP opcode/RGB order to avoid
; accidentally loading the wrong palette register by chance timing.
;
; C near-callable as: SetEDPAtLeft(int At  ibute, int Red, int Green,
;                                  int Blue, int ScanLine);

SEALparms struc
        dw      2 dup (?)       ;pushed BP & return address
Attribute dw    ?               ;DAC register to load (attribute for
                                ;  which to change color)
Red     dw      ?               ;red component to load
Green   dw      ?               ;green component to load
Blue    dw      ?               ;blue component to load
ScanLine dw     ?               ;scan line at start of which to put
SEALparms ends                  ;  EDP load

        public _SetEDPAtLeft
_SetEDPAtLeft proc      near
        push    bp              ;preserve caller's stack frame
        mov     bp,sp           ;point to local stack frame
        mov     ax,SCREEN_SEGMENT
        mov     es,ax
        mov     ax,SCREEN_WIDTH
        mul     ScanLine[bp]    ;start of scan line on which to load
        mov     bx,ax           ;ES:BX = start of scan line
        mov     al,byte ptr Attribute[bp]
        mov     es:[bx+5],al    ;DAC register to load
        mov     es:byte ptr [bx+1],191 ;EDP opcode
        mov     al,byte ptr Red[bp]
        mov     es:[bx+2],al    ;new red value
        mov     al,byte ptr Green[bp]
        mov     es:[bx+3],al    ;new green value
        mov     al,byte ptr Blue[bp]
        mov     es:[bx+4],al    ;new blue value
        pop     bp              ;restore caller's stack frame
        ret
_SetEDPAtLeft endp
        end
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

FIGURE 3: Screen layout in the sample application


Frame around whole screen drawn with attributes 6–190 and 224–255; wide enough to contain EDP sequences
EDP sequences to change the background color (attribute 1) on every scan line are stored along the entire right edge of the frame
Colors for top horizontally moving ball set here via EDP
Colors for middle horizontally moving ball set here via EDP
Colors for bottom horizontally moving ball set here via EDP
The background within the frame is drawn in attribute 1, which is changed on every scan line by the EDP sequences stored in the right edge of the frame
The horizontally moving balls are drawn in pixel-weighted combinations of attributes 2 and 3, with the colors of those attributes changed from ball to ball by EDP sequences in the left edge of the frame
The diagonally moving ball is drawn in pixel-weighted combinations of attributes 4 and 5

an EDP sequence could simultaneously be read to generate pixel data, you must ensure that the memory can never be in an intermediate state that could accidentally cause the wrong DAC register to be loaded. This one problem caused me much head scratching and gnashing of teeth, which I hope I've spared you.

ENABLING THE CEG/DAC

Enabling CEG mode is a simple matter of writing the characters "CEGEDSUN" to DAC register 222, followed by the number of the desired CEG/DAC mode. (See the CEG SDK documentation for a list of modes.) In CEG mode, bits 4–6 of the Mask register read back as the CEG/DAC version number, with at least one bit guaranteed to be 0. In non-CEG mode, or with a non-CEG DAC, bits 4–6 of the Mask register read back as the mask setting. Thus, CEG mode can be identified by enabling CEG mode as described above, then writing 0xFF to the DAC Mask register and immediately reading back the Mask register. If bits 4–6 are all 1-bits, it's not a CEG/DAC; otherwise, it is.

You will notice calls to **Delay** in **EnableCEGMode** in Listing 1. On my 8-MHz AT, **EnableCEGMode** would intermittently fail, even though a CEG/DAC was installed, until I put in delays between OUTs, perhaps because I have a prototype CEG/DAC. I recommend playing it safe and delaying between OUTs.

LOOKING BACK AND AHEAD

Was programming the CEG/DAC for animation as complex as I had feared? It *was* complex, but mostly because of learning to work with new technology, which is always a little mind-bending. In the end, the problems were solvable, and the results were definitely a cut above standard VGA.

In particular, EDP proved extremely valuable (once I had solved the problem of occasionally changing color 0 by chance timing). Pixel weighting has its uses but is frankly difficult to work with in an animation context; if I were doing this for a living, I'd focus on EDP. At any rate, I look forward to seeing the first commercial CEG/DAC-aware animation software, which should be like nothing else ever done on a $250 display adapter.

Less than 10 years ago, $250 wouldn't even buy a CGA! We've come a long way since then, and the adventure is just beginning. DJ

VENDOR INFORMATION

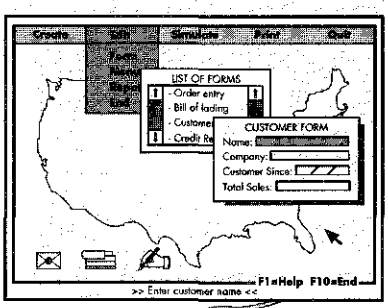
Edsun CEG/DAC (EL171, EL471, EL478)
Edsun Laboratories, Inc.
564 Main Street
Waltham, MA 02154
(617) 647-9300
Circle 165 on Reader Service Card

HI-SCREEN Pro II™

New Object Oriented Version For Powerful User Interfaces

Our new HI-SCREEN Pro II is an integrated environment for creating and managing user interfaces for your programs.

- Built-in editors allow you to interactively **generate objects** (windows, menus, icons, mouse cursors, graphs, help screens, etc.)
- Your programs can easily **manipulate** these **objects,** using a set of powerful functions.
- You can **test all objects** directly under the *editor in any graphic mode (Hercules, CGA, EGA, VGA).*
- New features include: New Icon Editor • New Menu Generator • New Graph Editor • New Library Generator • Enhanced scrolling (automatic scroll boxes) • 25/30/43/50 line mode support • Enhanced data validation • etc.



- Supports all versions of **C, Basic, Cobol, Fortran, Pascal, dBase, Assembly,** etc.

- Compatible with previous HI-SCREEN versions.
- Includes linkable modules for C, Pascal, Clipper and Quick Basic.

HI-SCREEN Pro II..... **$395**
Upgrades..... **Call**

30 day money-back guarantee • No royalties

1-800-338-2852
Tel.: 415-896-0708 • Fax: 415-896-0709

Softway, Inc.
185 Berry Street, Suite 5411,
San Francisco, CA 94107

CALL FOR DEMO

Circle 32 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Communicating over NetBIOS

Merge into the modem pool's fast lane with Turbo Pascal

By Brett Glass

The financial gurus all seem to be crying "recession," and many companies are taking belt-tightening measures. On the computer front, an area ripe for cost-cutting will be upgrades and peripherals—especially modems.

Installing a 2400-baud modem is more expensive than the $120 or so that the hardware costs. There's the matter of adding a phone line for it, which can get especially tricky on digital PBXes. Consider also the wiring costs, installation labor, and another $100 or so if you want a dedicated business phone line. The total expense clearly enters the realm of serious money.

That's why corporate computer users are probably going to see an increase in the number of modem pools. A modem pool is a group of modems—usually attached to a computer in a telephone closet—that can be shared by users on a LAN. Modem pools have their problems, but they also provide access for a fraction of the cost of a shiny new modem.

SLOW TURNAROUND

The main advantage of modem pools is their savings, and their biggest disadvantage is delays. IBM configured the original PC's BIOS to limit communications speeds to 9600 baud or less. At higher rates, there was a good chance that characters would be dropped. The reason for the limitation wasn't inefficiency in the BIOS code (which was very simple); rather, IBM was concerned that the processor might not have the "oomph" to field all of those characters at higher speeds.

Now consider the LAN situation. A LAN is much faster than a serial port, so raw throughput isn't a problem. Instead, the most serious problem is overhead: building, encoding, and sending the packets (even if they contain several serial characters) can involve many times the work of writing to a serial port. Thus, communications via a modem server still tend to suffer from disconcertingly long character turnaround times—a significant delay between the time you hit a key and the time it is echoed. But performance is usually fine for uploads and downloads as long as you use a "windowing" protocol (such as ZMODEM or CompuServe B+) to keep delayed acknowledgements from hurting your transfer rates.

LINKING ACROSS THE LAN

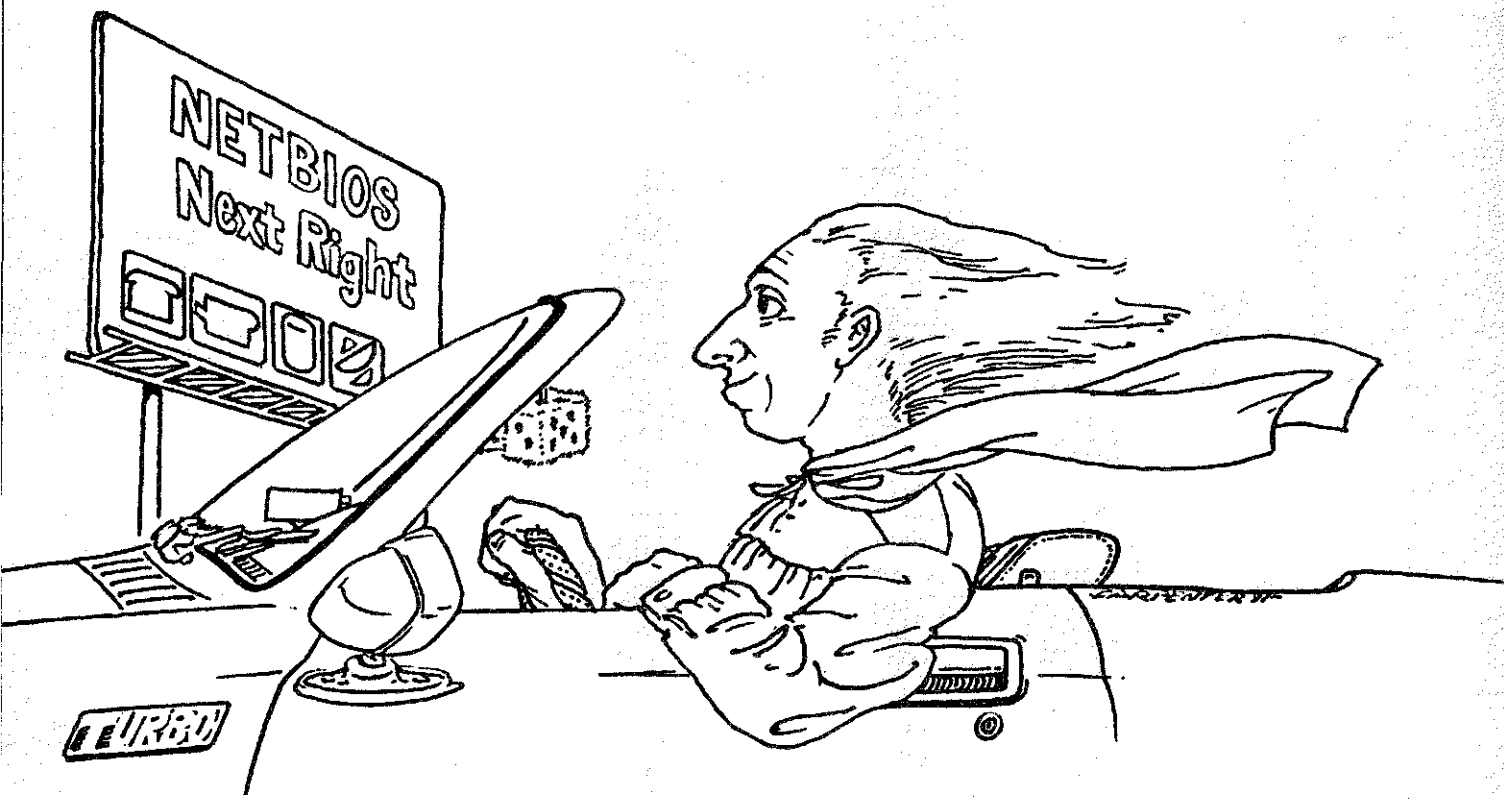
One of the knottiest problems that a LAN must handle is the interface between the communications software (say, CrossTalk, TAPCIS, or Procomm) and the remote modem. Had IBM designed its PC with more forethought, this wouldn't be a problem; there's a well-defined API that could hide the specifics of the hardware and make it easy to write a communications program. This is one area where the Macintosh and even Unix outperform the PC: the IBM PC BIOS provides a less-than-ideal interface. It insists on doing its own peculiar form of handshaking, which can disconnect or reset a modem. It refuses to transmit characters if the Carrier Detect line is off (making it necessary to dial a "smart" modem). Worst of all, it isn't interrupt-driven—it can drop characters even if the software polls it assiduously. For this reason, all quality communications software goes right to the hardware and takes over the UART chip.

This means that a LAN communications server needs help from the communications program, usually in one of two ways. First, the communications program can "force" itself to use the primitive BIOS services (provided through INT 14h), trusting the server to buffer characters so they don't get dropped. The advantage of this approach is simplicity: there are only four BIOS functions to know about (or six if you use the "enhanced" set provided by the PS/2 line), and they're all clearly documented in many reference books.

However, because the INT 14h functions weren't designed for modem pools,

Brett Glass is a freelance writer, programmer, and consultant residing in Palo Alto, California. He can be contacted at rogue@well.sf.ca.us on Internet, on CompuServe at 72267,3673, or as glass on Bix.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



some desirable features aren't there. For instance, there's no provision to "open" and "close" a serial port; if users are to share the modems, some special mechanism must be devised to let them connect to and disconnect from a port. Nothing in the BIOS interface lets you specify which modem you want to use or find out which ones are available. Still, because the INT 14h interface is so well known, many LAN and communications software vendors support it. There's one other advantage: if you want to use your network communications server to capture output from an existing program that uses INT 14h, you can do so without changing anything. For instance, if you put a serial printer on your communications server, you can then print to it by using the DOS MODE command to redirect printer output.

The second approach is to use a different software interface to the modem server. There are two popular alternative interfaces: the NetWare Advanced Service Interface (NASI) was originally developed by Network Products Corporation for Novell; Network Products calls their version of the same API the Network Communications Services Interface (NCSI) (see Vendor Information, p. 42). NASI/NCSI use INT 6Bh, and the basic functions are mimicked by the Ungermann-Bass Network Interface (UBNI). The IBM Asynchronous Communication Services Interface (ACSI), which uses NetBIOS over Token Ring or the IBM PC Network, is most commonly used by corporations that have large investments in IBM networks and mainframes. Both interfaces have the functions that INT 14h lacks. However, literature on them is harder to track down, and they're not as widely supported. Some vendors have also added in their own advanced functions to INT 14h, but these functions aren't standardized.

SERVER PROTOTYPE

For the project described in this article, I wanted to prototype a communications server that would work with the widest possible variety of LANs and communications software. I therefore chose to implement the INT 14h interface (with the PS/2 extensions) and to use NetBIOS calls to interface to the LAN. NetBIOS has three key advantages over IPX and other proprietary protocols. First, every PC LAN uses or emulates it. Second, it's peer-to-peer (the communications server need not be a LAN server). Finally, it's available as a layer on top of TCP/IP, making it theoretically possible to have a modem server in another city across a nonproprietary WAN (imagine the phone bills you could save!).

I began by designing a test program, a simple terminal emulator that uses only INT 14h calls. This proved to be an excellent debugging tool. Because I had the source, I could run the emulator under Turbo Debugger and watch its every interaction with the communications server. I then started on the meat of the project: a small TSR to run in the client workstation, and a larger server program to provide the communications services.

NETBIOS OBJECTS

I built in NetBIOS support via an object-oriented NetBIOS unit that treats Network Control Blocks (NCBs) as objects. To access NetBIOS, you create NCBs, fill in the appropriate fields, and tell them what to do. The resulting code is easy to follow, especially if you're using "with" statements to shorten the names. The NetTools unit handles the most common network operations. Together, these two units do the bulk of the work for the client and the server. They are available (along with the test program) on this issue's code disk or on *PJ*'s BBSs (see p. 95).

Because a client will want to connect to the server for some period of time, I opted to use NetBIOS sessions—rather than datagrams—to establish the link. This approach has pros and cons: sessions make it easy to send and receive messages, but they can get cut off by adverse network conditions and require more code than datagrams. I chose to use sessions because they provide automatic confirmation of delivery; you don't have to worry about a message getting lost.

DIVISION OF LABOR

Another design consideration was how to split the work load between client and server. Because a workstation might put a

ILLUSTRATION BY JIM CARPENTER

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Multitasking Real-Time

Do the multitasking systems you are considering using seem a bit like overkill or need an awful lot of memory? Maybe all you really need is a system which can run more than one DOS program concurrently. If this sounds like your situation, consider:

MultiDos Plus™

Flexible preemptive scheduling of multiple tasks with full access to MS-DOS or PC-DOS. Supports the loading and execution of tasks in LIM 4.0 expanded memory. For more information please call for a free brochure.

Call our BBS for free demo software.

The complete software package is only $99 + S/H. Visa/MasterCard accepted.

30-day money back guarantee.
(subject to a $15 restocking fee)

Nanosoft Inc.
13 Westfield Road
Natick, MA 01760
800-678-2141
508-651-0091
FAX 508-655-8860
BBS 508-650-9552

Circle 33 on Reader Service Card

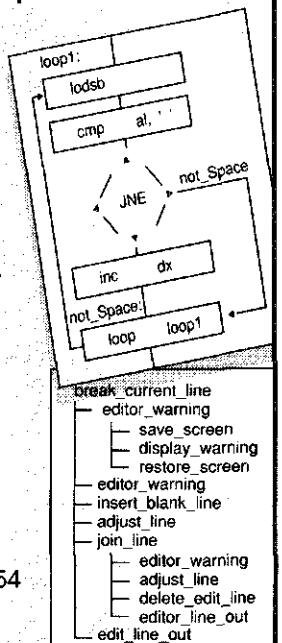
ASMFLOW PROFESSIONAL

YOU CAN STEP BACK AND TAKE A LOOK AT YOUR CODE

$199.95 S/H $5.00 Outside U.S. add $10.00

FLOW CHART AND ANALYZE YOUR ASSEMBLY LANGUAGE SOURCE CODE

- Flow Charts
- Tree Diagrams
- Stack Sizing
- Register Analysis
- CPU Timing Analysis
- Procedural X-Reference
- 8088/87 to 80486
- Context-Sensitive Help
- MASM 5.1 Compatible
- Macro Expansion
- 8085, Z-80 and 8051 Versions



QUANTASM CORPORATION
19855 Stevens Creek Blvd, Suite 154
Cupertino, CA 95014
(408) 244-6826 – Free Demo

VISA • MC 30 – DAY MONEY BACK GUARANTEE

Circle 34 on Reader Service Card

LISTING 1

```
{
** Communications Client Version 1.0                    May 1, 1991 **
** Copyright 1987,1988,1991 by L. Brett Glass, Systems Consultant
}
program Client; {Client TSR}
uses NetBIOS,NetTools,DOS;

{$M 8192,0,0} {Use 8K of stack, no heap}
{$R-}
{$S-}
{$I-}       {Turn off space- and time-consuming options}
{$V-}

type
  PortNumType = 1..4;

type
  CommRegType = record
    case Boolean of
      TRUE: (dx,cx,bx,ax : Word);
      FALSE: (dl,dh,cl,ch,bl,bh,al,ah : Byte);
    end;
var
  myPermNodeName, remotePortName : NetName;
  biosPortNum : PortNumType;
  commSessionNum : Byte;
  oldInt14 : Pointer;
  clientSendBlock,clientRcvBlock : NCB;
  saveStackSeg : Word;
  saveStackPtr : Word;

const
  ERRORFLAG = $80; {Flag to indicate comm timeout or error}

procedure ValidateParms;
  var
    tempStr : String;
  begin
  if ParamCount <> 2 then
    begin
    Writeln('Usage: CLIENT <Network Name> <Port Number>');
    Writeln(' E.G.: CLIENT MyModem 2');
    Writeln('       Connects COM2: to a remote networked');
    Writeln('        communications port with the name MyModem');
    Halt
    end;
  tempStr := ParamStr(1);
  if (Length(tempStr) = 0) or (Length(tempStr) > 16) then
    begin
    Writeln('Error: Serial port/modem name must be 1 to 16 characters');
    Halt
    end;
  FillChar(remotePortName,SizeOf(remotePortName),' ');
  Move(tempStr[1],remotePortName[1],Length(tempStr));
  tempStr := ParamStr(2);
  if (Length(tempStr) <> 1) or (tempStr[1] < '1') or
    (tempStr[1] > '4') then
    begin
    Writeln('Error: Port number must be 1 through 4');
    Halt
    end;
  biosPortNum := Ord(tempStr[1])-Ord('1'); {BIOS uses 0=COM1, etc.}
  end; {ValidateParms}

procedure Connect;
  begin
  if not NetBIOSPresent then
    begin
    Writeln('Error: NetBIOS not available');
    Halt;
    end;
  if NetToolsGetMyName(myPermNodeName) <> GOOD_RTN then
    begin
    Writeln('Error: Unable to determine adapter status');
    Halt;
    end;
  {Do a call. Set up a session with five-second timeouts.}
  if NetToolsCall(myPermNodeName,remotePortName,10,10,commSessionNum) <>
GOOD_RTN then
    begin
    Writeln('Error: Unable to establish session with server');
    Halt;
    end;
```

continued

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

premium on memory and would want the most compact client software possible, I made the client (Listing 1) very "dumb": it simply takes over INT 14h and watches for requests involving the remote serial port. When it finds such a request, it packs up the relevant registers and sends them to the server for processing. Were I converting this prototype into a commercial product, I'd also use other TSR space-saving techniques: I'd release my environment, put global data at the beginning, and leave initialization code at the end of the program so that it could be removed after it was used. However, this isn't possible with Turbo Pascal's memory model, so the prototype is larger than most commercial client programs.

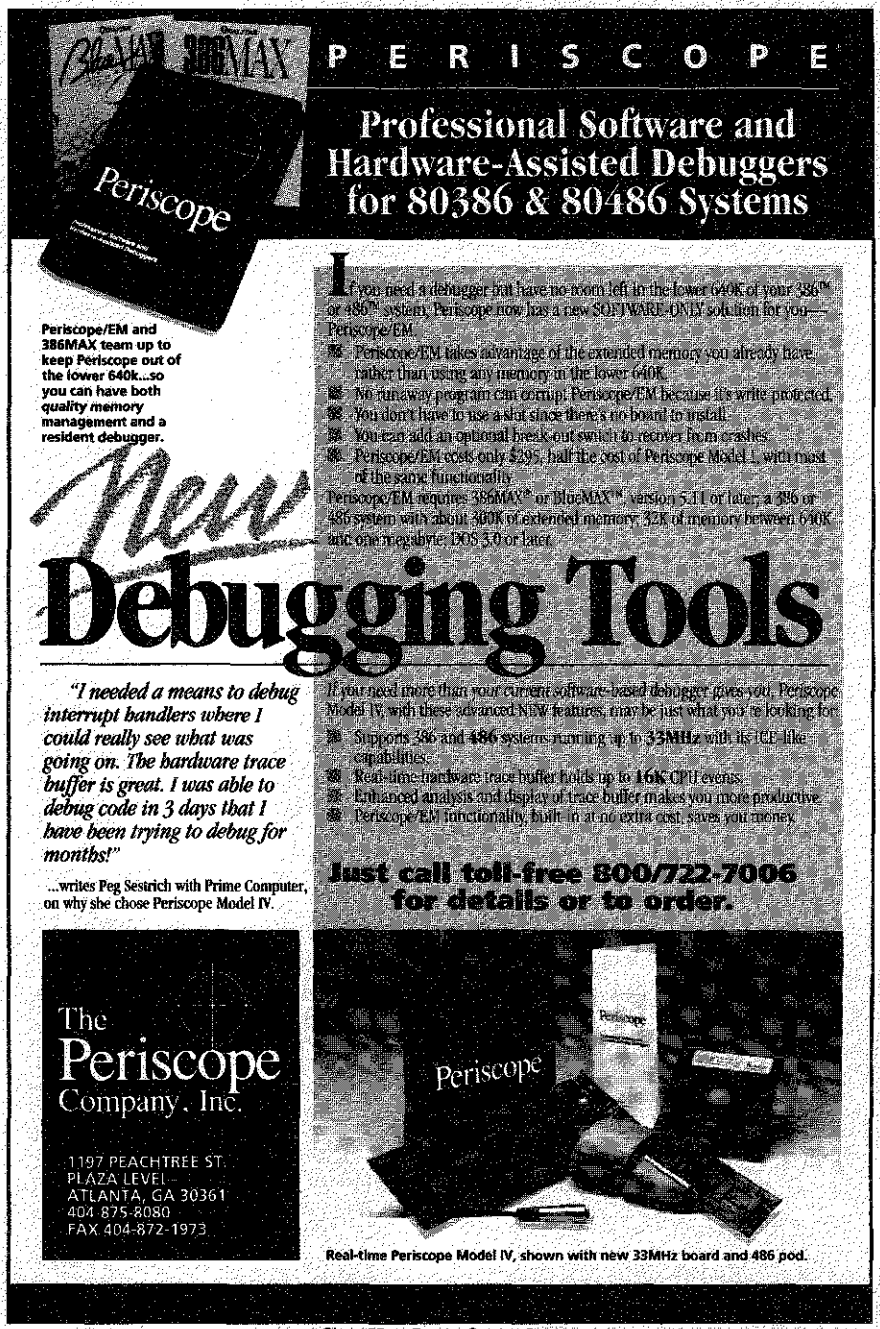
To invoke the client, type CLIENT <Network Name> <Port Number>. "Network Name" is the name of the server serial port, and "Port Number" is the port number it will have on the local machine (1 for COM1:, 2 for COM2:, etc.). This prototype client doesn't have an uninstall facility; it uses the specified port until the machine is rebooted or until the TSR is removed by a Mark/Release program.

The server (Listing 2) is more complex. It uses interrupts to handle incoming characters and maintains a circular buffer to prevent their getting lost. Note that the receive buffer is a "fall out" buffer—when it overflows, it loses the oldest character rather than the newest. If characters are received on a port that currently has no client, the next client that connects to the port can see the most recent input. This version of the server does not do interrupt-driven transmission, although it would be simple to add. Also, to keep it simple, this prototype server handles only one serial port. To add more, you would need to make arrays out of the global variables that hold information about each port.

This version of the server isn't a TSR. Although it's possible to make it one (and third-party software toolkits such as TesSeRact make it downright easy), my experience indicates that it may not be a good idea, especially if the server is handling even one high-speed modem. A server that functions as a TSR has to split its time with other programs on the machine, possibly delaying responses until NetBIOS is not busy (you can't send a reply from within a NetBIOS "post" routine) and a timer tick interrupt arrives to wake the server up. Even if you speed up the timer (it normally ticks only about 20 times a second) or use the real-time clock interrupt (which occurs about a thousand times a second), you may find that the server doesn't do a good job of keeping up.

LITTLE WHITE LIES

When I implemented the server, I found I had to make it "lie" to the client in order to deal with the subtleties of the INT 14h interface. For instance, because INT 14h's Subfunction 2 (Receive Character) waits a fixed amount of time for a character before giving up, communications programs usually call Subfunction 3 (Get Status) first to confirm that a character is present. However, although the non–interrupt-driven BIOS can read two UART registers and report them verbatim as the status, the interrupt-driven server can't. (It's constantly clearing characters out of the UART in response to interrupt


PERISCOPE
Professional Software and Hardware-Assisted Debuggers for 80386 & 80486 Systems
Periscope
Periscope/EM and 386MAX team up to keep Periscope out of the lower 640k...so you can have both quality memory management and a resident debugger.
If you need a debugger that have no room left in the lower 640K of your 386™ or 486™ system, Periscope now has a new SOFTWARE-ONLY solution for you—Periscope/EM.
Periscope/EM takes advantage of the extended memory you already have, rather than using any memory in the lower 640K.
No runaway program can corrupt Periscope/EM because it's write protected.
You don't have to use a slot since there's no board to install.
You can add an optional break-out switch to recover from crashes.
Periscope/EM costs only $295, half the cost of Periscope Model I, with most of the same functionality.
Periscope/EM requires 386MAX® or BlueMAX™, version 5.11 or later, a 386 or 486 system with about 300K of extended memory, 32K of memory between 640K and one megabyte, DOS 3.0 or later.
New
Debugging Tools
"I needed a means to debug interrupt handlers where I could really see what was going on. The hardware trace buffer is great. I was able to debug code in 3 days that I have been trying to debug for months!"
...writes Peg Sestrich with Prime Computer, on why she chose Periscope Model IV.
If you need more than your current software-based debugger gives you, Periscope Model IV, with these advanced NEW features, may be just what you're looking for.
Supports 386 and 486 systems running up to 33MHz with its ICE-like capabilities.
Real-time hardware trace buffer holds up to 16K CPU events.
Enhanced analysis and display of trace buffer makes you more productive.
Periscope/EM functionality built in at no extra cost, saves you money.
Just call toll-free 800/722-7006 for details or to order.
The Periscope Company, Inc.
1197 PEACHTREE ST.
PLAZA LEVEL
ATLANTA, GA 30361
404-875-8080
FAX 404-872-1973
Periscope
Real-time Periscope Model IV, shown with new 33MHz board and 486 pod.
Circle 35 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
  {Initialize NCBs for the conversation}
  clientSendBlock.Init(SEND);
  clientRcvBlock.Init(RECEIVE);
  end; {Connect}

procedure Int14Handler(flags,cs,ip,ax,bx,cx,dx,si,di,ds,es,bp : Word);
  interrupt;
  {All Int 14 requests are sent off to the server for processing.
   The server takes care of queueing, returning status, etc.
   Note that this code doesn't filter out invalid function codes;
   this allows us to add functionality at the server without
   changing the basic client.}
  var
    commRegs : CommRegType absolute dx;
  begin
  if dx <> biosPortNum then {Chain to old vector}
    asm
    mov sp,bp
    pop bp; pop es; pop ds; pop di; pop si;
    pop dx; pop cx; pop bx; pop ax;
    jmp oldInt14;
    end
  else
    begin
    asm sti end; {Turn on interrupts}
    with clientSendBlock do
      begin
      bufPtr := Addr(commRegs);
      lsn := commSessionNum;
      len := SizeOf(CommRegType);
      if ReturnCode = GOOD_RTN then
        with clientRcvBlock do
          begin
          bufPtr := Addr(commRegs); {Point to block with dx,cx,bx,ax}
          len := SizeOf(commRegs); {Four words long}
          lsn := commSessionNum;
          if (ReturnCode <> GOOD_RTN) or (len <> SizeOf(commRegs)) then
            commRegs.ah := retcode or ERRORFLAG; {Slip in NetBIOS
                                                  error code}
          end
      else {If send failed}
        commRegs.ah := retcode or ERRORFLAG;
      end
    end {case ah}
  end; {Int14Handler}

procedure DivertInt14;
  begin
  GetIntVec($14,oldInt14);
  SetIntVec($14,Addr(Int14Handler));
  end; {DivertInt14}

begin {Client}
ValidateParms; {Check and parse command line}
Connect;
DivertInt14; {Capture INT 14 so we can provide services}
SwapVectors; {Prepare to go resident}
Keep(0); {Go resident}
end. {Client}
```

LISTING 2

```
{*
** Communications Server Version 1.0                    May 1, 1991
**
** Copyright 1987,1988,1991 by L. Brett Glass, Systems Consultant
**
}

program Commserver;
{$M 8192,0,0} {Use 8K of stack, no heap}

uses NetBIOS,NetTools,DOS,CRT;

{BIOS list of UART base I/O addresses}

type
  PortNumType = 1..4; {We restrict comm port numbers to 1 thru 4
                       because that's what's in the BIOS table}
var
  biosPortTable : array[portNumType] of Word absolute $40:00;

{UART declarations}

{The following constants give the offsets of ports from
 the UART's base address}

const
  RBR = $0;     {Receiver Buffer Register}
  THR = $0;     {Transmit Holding Register}
  DLL = $0;     {Low byte of divisor}
  DLH = $1;     {High byte of divisor}
  IER = $1;     {Interrupt Enable Register}
  IIR = $2;     {Interrupt Identification Register}
  LCR = $3;     {Line Control Register}
  MCR = $4;     {Modem Control Register}
  LSR = $5;     {Line Status Register}

{The following table lists baud rates. It maps the codes used
 in the BIOS to divisors.}

const
  divisorTable : array[0..8] of Word = (
    {110} $417, {150}  $300, {300}  $180, {600} $0C0, {1200} $060,
    {2400} $030, {4800} $018, {9600} $00C, {19200} $006);

{The following constants are necessary for managing interrupts}

const
  OCW1  =  $21; {Port address of enable bits for 8259}
  OCW2  =  $20; {Port address for commands to 8259}
  NSEOI =  $20; {Nonspecific EOI command}

  {Bit to use to mask interrupts at 8259}
  intMask : array [PortNumType] of Byte = ($10,$08,$10,$08);

  {Vector numbers for ports}
  commIntVec : array [PortNumType] of Byte = (12,11,12,11);

{The following masks are useful to manipulate bits
  in the UART registers}
const
  ERBFI = $01;  {Mask to enable receive interrupts}
  THRE =  $20;  {Mask for THRE}
  DTR = $01;    {Mask for DTR}
  DLAB = $80;   {Mask for DLAB}
  RTS = $02;  {Mask for RTS}
  OUT2 = $08; {Mask for OUT2}

{The following constants are handy to intialize the UART}

const
  ONESTOP = $00;{Mask for 1 stop bit}
  NOPARITY = $00; {Mask for  no  parity}
  EIGHTBITS = $03;{Mask for 8 bits/char}

{The following constant is returned in ah to indicate that no chars
 are available during a read}

  ERRORBYTE = $80;
{The following byte is used to mask the line status on a successful
 read. Note that the uppermost bit isn't allowed through, since
 a successful read does not set the timeout bit.}

  READSTATUSMASK = $0F;

{The following byte is used to mask requests to change the serial
 port parameters.}

  CHARMASK = $1F;

{These constants determine the initial baud rate for the port,
 the size of the receive buffer, and other serial port parameters}

const
  INITIALBAUD = 4; {Start at 1200 baud when initializing}
  BUFFMAX = 255;  {Size of receive buffer -- should be 2^n - 1}

type
  CommRegType = record
    case Boolean of
      TRUE: (dx,cx,bx,ax : Word);
      FALSE: (dl,dh,cl,ch,bl,bh,al,ah : Byte);
    end;

var
  portName : NetName;     {Network name of serial port}
  portNameNum : Byte;     {Number of this name in local name table}
  portNum : PortNumType; {Serial port number}
  portBase : Word;        {Base address of UART}
```

continued

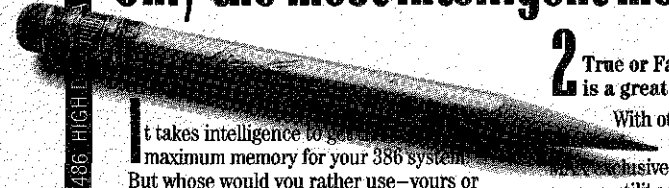
PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
  rcvBuff : array [0..BUFFMAX] of Byte; {Circular fall-out buffer}
  buffIn, buffOut : Integer; {Buffer head and tail pointers}
  oldIntVec : Pointer;   {Storage for old interrupt vector}
  commSessionNum : Byte; {Number of NetBIOS session}
  clientName : NetName;  {Name of client}

function HexStr(var num; byteCount : Byte) : String;
  const
    hexChars : array [0..$F] of Char = '0123456789ABCDEF';
  var
    numArray : array [Byte] of Byte absolute num; {Access bytes of num}
    tempStr : String; {Holds result}
    tempLen : Byte absolute tempStr; {Length of result}
  begin
  tempLen := 0;
  for byteCount := Pred(byteCount) downto 0 do {numArray is 0-based}
    tempStr := tempStr +                        {Add:    }
               hexChars[numArray[byteCount] shr 4]  + {Hi digit}
               hexChars[numArray[byteCount] and $F];  {Lo digit}
  HexStr := tempStr
  end; {HexStr}

procedure ValidateParms;
  var
    tempStr : String;
  begin
  if ParamCount <> 2 then
    begin
    Writeln('Usage: COMMSERVER <Network Name> <Port Number>');
    Writeln(' E.G.: COMMSERVER MyModem 2');
    Writeln('       Makes COM2: a networked communications port');
    Writeln('        with the name MyModem');
    Halt
    end;
  tempStr := ParamStr(1);
  if (Length(tempStr) = 0) or (Length(tempStr) > 16) then
    begin
    Writeln('Error: Serial port/modem name must be 1 to 16 characters');
    Halt
    end;
  FillChar(portName,SizeOf(portName),' ');
  Move(tempStr[1],portName[1],Length(tempStr));
  tempStr := ParamStr(2);
  if (Length(tempStr) <> 1) or (tempStr[1] < '1') or
    (tempStr[1] > '4') then
    begin
    Writeln('Error: Port number must be 1 through 4');
    Halt
    end;
  portNum := Ord(tempStr[1])-Ord('0');
  portBase := biosPortTable[portNum];
  if portBase = 0 then
    begin
    Writeln('Error: Port does not exist');
    Halt
    end;
  end; {ValidateParms}

procedure IntHandler; interrupt; {Received character ISR}
  begin
  asm sti end;
  buffIn := Succ(buffIn) and BUFFMAX;
  rcvBuff[buffIn] := Port[portBase+RBR];
  if buffIn = buffOut then {Queue is overflowing. Keep newest characters.}
    buffOut := Succ(buffOut) and BUFFMAX;
  Port[OCW2] := NSEOI
 end; {IntHandler}
```

continued

640K WINDOWS ROM LIM EMS HMA DOS 5.0 XMS BUFFERS UMB 386MAX 486SX DRIVERS 386SX NETWORKS TSRs PAGEFRAME IPX INITIALIZE RAM MEMORY BIOS CACHE VGA MOUSE 386 BLUEMAX 486 HIGH

Only the most intelligent memory managers can pass this test.



It takes intelligence to get the maximum memory for your 386 system. But whose would you rather use—yours or your memory manager's? Here's a little quiz to help you make the smart choice.

1 True or False—All memory managers are alike.

False. Most memory managers free up space for applications by moving TSRs and device drivers from conventional memory into high DOS memory. With other products, you could spend days guessing about the best way to do that. But MAX uses its intelligence to find the best possible fit—automatically. Guaranteed. And because many TSRs need much more space to load than they need to run, MAX has an exclusive feature that "borrows" up to 64K of high DOS for loading, so it can pack even more in. With MAX, you always get the most memory possible.

2 True or False—Using TSRs in Windows is a great way to crash your system.

With other products, that's definitely true. But thanks to another MAX exclusive, you can use nearly any pop-up utility as many times as you want.

3 With DOS 5.0, there's no need for a memory manager—right or wrong?

That depends. DOS 5 does free up some memory with its smaller program size. And it does let you place programs into high DOS manually. But for optimal memory management, you need MAX—the powerful, automated, full-service program that specializes in memory. So while DOS 5 is good, DOS 5 with MAX is outstanding.

The answer is easy. The delivery is free.

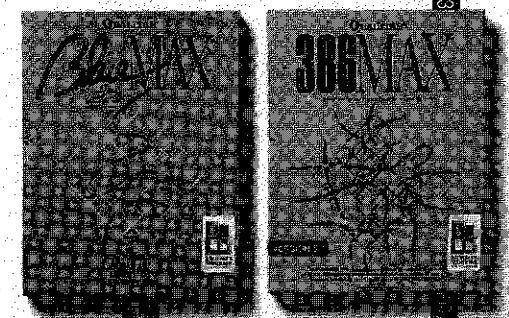
Why not graduate to the Intelligent Memory Managers? There's 386MAX® for any IBM compatible 386 or 486 based system at just $130. And BlueMAX™ at $155, the only way to get 250% more high DOS memory on all 386 and 486 IBM PS/2s.

Order direct before 3:00 pm Eastern Time, and overnight shipping is free.* We'll even send a free copy of ASQ,™ the intelligent memory analyzer.

See your dealer or call toll-free today.

1-800-676-6386 ext. 824

FEDERAL EXPRESS



The Intelligent Memory Managers™

640K WINDOWS ROM LIM EMS HMA DOS 5.0 XMS BUFFERS UMB 386MAX 486SX DRIVERS 386SX NETWORKS TSRs PAGEFRAME IPX INITIALIZE RAM MEMORY BIOS CACHE VGA MOUSE 386

© 1991 Qualitas. Qualitas, 7101 Wisconsin Avenue, Suite 1386, Bethesda, MD 20814. *All company and product names are trademarks or registered trademarks of their respective owners.* **System Requirements: Any 386 or 486 PC or PS/2, min. 256KB of extended memory, DOS 3.0 or higher, high density floppy disk or hard disk drive.**

QUALITAS®

**Offer valid in North America only.*

Circle 36 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

requests, so those registers would never be able to show that a received character is available.) Thus, this server "fakes" the bit that indicates whether a character has come in. Also, to avoid delays, the server returns an "instant" timeout if it's asked to read an incoming character when none is present. Finally, the server doesn't use the handshaking sequences used by the BIOS. This allows it to work with modems that have carrier detection enabled and prevents conversations from being cut off when the BIOS drops DTR.

TIME FOR THE ACID TEST

After I got the client and server working with my small terminal program, I subjected them to the "acid test": a commercial communications package. I installed CrossTalk Mk4 (an expensive, powerful, and somewhat intimidating program) and connected its "INT14" driver to make it use the network. Mercifully, the program was able to call CompuServe and log in successfully. Cursor movement was sluggish, but I had no problems controlling the modem or downloading a file.

The performance of these initial attempts was acceptable—comparable to the free communications server software that came with my LAN package—but I'm convinced I can do better. Future experiments will include tricks such as sending characters to the client immediately (rather than waiting for requests) and transferring more than one character at a time in each direction. I'm also considering rewriting the client in assembly language and giving it a "client control panel" that will let it pick a port, disconnect from one, or uninstall itself on the fly.

The process of writing a communications server for modems was instructive, but a modem pool is a poor substitute for what most people really want: a good, fast modem on their desks. I'm looking forward to the day when I can create a communications server that connects to the outside world without using a modem at all. With the advent of ISDN, it should eventually be possible to make a server that hooks a LAN directly to the digital communications lines used by your phone. Imagine: no line noise, no busy signals, and a 64-kilobits/sec link to the outside world. Now *there's* a good reason to do communications over a LAN. PJ

VENDOR INFORMATION

NCSI
Network Products Corporation
1440 West Colorado Boulevard
Pasadena, CA 91105
(800) 638-7765
Circle 167 on Reader Service Card

NASI
Novell Development Products
5918 West Courtyard Drive, # 170
Austin, TX 78730-5036
(800) 638-9273
Circle 168 on Reader Service Card

AMX

REAL-TIME MULTITASKING KERNEL

8086/88, 80x86/88 80386
Z80, 64180, 8080/85 68000/10/20

- Fast, reliable operation
- Compact and ROMable
- PC peripheral support
- DOS file access
- C language support
- Preemptive scheduler
- Time slicing available
- Configuration Builder
- Complete documentation
- Intertask messages
- Message exchanges
- Dynamic operations
  - task create/delete
  - task priorities
  - memory allocation
- Event Manager
- Semaphore Manager
- List Manager
- InSight™ Debugging Tool

THE BEST
Join over 600 developers such as IBM®, Xerox, Hewlett Packard, Hayes, Hughes Aircraft and NASA.

CHOOSE AMX
The best low-cost, high-performance real-time multitasking system available today.

No Royalties
Source Code Included

Demo Disk and Manual only $75 US *Call for prices for*
AMX 86 $3000 US *other processors.*
(Shipping/handling extra)

IBM is a registered trademark of IBM Corp.
Z80 is a trademark of Zilog, Inc.
AMX, AMX 86, inSight are trademarks of KADAK Products Ltd.

KADAK Products Ltd.
206-1847 West Broadway
Vancouver, B.C., Canada
V6J 1Y5
Telephone: (604) 734-2796
Fax: (604) 734-8114

Circle 37 on Reader Service Card

Imagine...

Information on Demand

Imagine having the information you want at your fingertips. Imagine that it's free.

Programmer's Journal Reader Service Cards — where imagination and reality come together.

Valuable information on products and services can be yours for **FREE** — we even pay the postage! Just tear out the *PJ* Reader Service Card opposite the Ad Index on p. 80. Circle the numbers from the ads or editorial mentions of the items you want to know more about.

Now, imagine yourself filling out the card and sending it in . . . today.

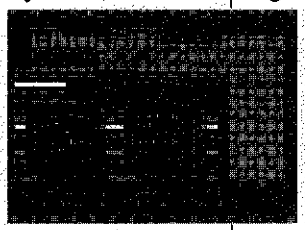
PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

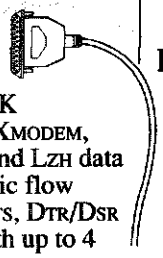
```
procedure InitPort;
  begin
  buffIn := 0;
  buffOut := 0;
  Port[portBase+IER] := 0;  {Disable comm interrupts first}

  {Hook into the interrupt vector}
  GetIntVec(commIntVec[portNum],oldIntVec);
  SetIntVec(commIntVec[portNum],Addr(IntHandler));

  {Initialize the UART}
  Port[portBase+LCR] := Port[portBase+LCR] or DLAB; {Access divisor latch}
  Port[portBase+DLL] := Lo(divisorTable[INITIALBAUD]); {Set baud rate
                                                        to 1200 }
  Port[portBase+DLH] := Hi(divisorTable[INITIALBAUD]);
  Port[portBase+LCR] := EIGHTBITS or ONESTOP or NOPARITY; {Clear DLAB and
                                                           set  parms}

  Port[portBase+MCR] := DTR or RTS or OUT2; {Enable interrupts, turn on
                                             DTR & RTS}

  {Turn  on  interrupts at the 8259}
  Port[OCW1] := Port[OCW1] and not(intMask[portNum]);
  if Port[portBase+LSR] <> 0 then; {Clear errors}
  if Port[portBase+RBR] <> 0 then; {Remove any trash in RBR}
  Port[portBase+IER] := ERBFI; {Enable UART receive interrupts}
  end; {InitPort}

procedure Shutdown;
  begin
  Port[portBase+IER] := 0; {Kill UART interrupts}
  Port[portBase+OCW1] := Port[portBase+ OCW1]
    or intMask[portNum]; {Mask interrupts at PIC}
  Port[portBase+MCR] := 0; {Shut off DTR, RTS, OUT2}
  SetIntVec(commIntVec[portNum],oldIntVec);
  if NetToolsDeleteName(portName) <> GOOD_RTN then; {Only try once}
  Writeln('Communcations server shutting down....');
  Halt;
  end; {Shutdown}

function UserAbort : Boolean;
  begin
  UserAbort := FALSE;
  if KeyPressed then
    case ReadKey of
      #3 : UserAbort := TRUE;
      #0 : if KeyPressed then
        UserAbort := (ReadKey = #0)
    end;
  end; {UserAbort}

procedure AwaitClient;
  var
    listenBlock : NCB;

  procedure ListenError;
    begin
    Writeln('Error: NetBIOS error when listening for clients');
    Shutdown
    end; {ListenError}

  begin
  case NetToolsStartListen(listenBlock,wildName,portName,10,10) of
    GOOD_RTN,COMMAND_PENDING:;
  else
    ListenError
  end;
  while TRUE do
```

continued

OOP Async Tools for Turbo Pascal 6.0 with ZMODEM and Data Decompression

Async Professional is a full-featured object-oriented communications toolkit that enables you to build powerful async applications faster and easier. Async Professional has all the standard features you'd expect of an async toolbox. Plus the robust ZMODEM transfer protocol, data decompression tools, and both object-oriented and procedural calling interfaces.





You get ■ interrupt driven buffered I/O to 115K baud ■ ZMODEM, Kermit, XMODEM, YMODEM protocols ■ ZIP and LZH data decompression ■ automatic flow control–XON/XOFF, CTS/RTS, DTR/DSR ■ COM1- COM8 support with up to 4 ports open at once ■ 16550 buffered UART support ■ hardware interrupt sharing on PS/2 ■ ANSI terminal emulation ■ modem control including Hayes, V.32, V.42, and MNP5.

Hot Example Programs Too!

Demo programs include a powerful multi-window comm program with menus, text editor, and mouse support, built using Object Professional, and a simple comm program that depends on no other products.

Powerful Async Debugging

Event logging and tracing options create a time-stamped audit trail of all async interrupts and characters sent or received. You'll debug your application faster without the need for special debugging hardware.

Full Source, Documentation

Async Professional includes complete documentation, pop-up help, free technical support on CompuServe and by telephone direct from the authors. You get full source code and pay no royalties.

"Async Professional is an excellent series of routines...and very complete. It's better than anything I've seen in Pascal, C or assembler."

Richard Wilkes, Author, TAPCIS

Async Professional, only $139.

Call toll-free to order.
1-800-333-4160

Satisfaction guaranteed or your money back within 30 days. Add $5 per order for standard shipping in U.S./Canada. Inquire about other shipping options. Includes both 5.25" and 3.5" disks. Registered LiteComm owners call for special upgrade offer. OOP units require Turbo 5.5 or 6.0; non-OOP units require Turbo 5.0 or later.

TURBO Power

9AM–5PM MST Monday through Friday, USA & Canada.
For more information call (719) 260-6641, fax to (719) 260-7151, or send mail to CompuServe ID 76004,2611
TurboPower Software PO Box 49009 Colorado Springs, CO 80949-9009
© TurboPower Software 1991

Circle 38 on Reader Service Card

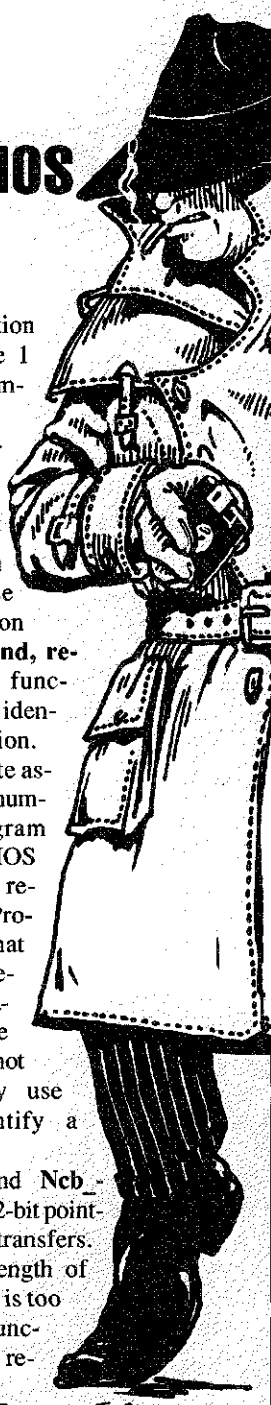
PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
    begin
    if UserAbort then
      begin
      NetToolsAbortListen(listenBlock);
      Shutdown
      end;
    case NetToolsCheckListen(listenBlock,commSessionNum,clientName) of
      GOOD_RTN : Exit;
      COMMAND_PENDING:;
    else
      ListenError
    end
    end;
  end; {AwaitClient}

function CharAvail : Boolean;
  begin
  asm cli end;
  CharAvail := (buffIn <> buffOut); {Do test with interrupts off}
  asm sti end;
  end; {CharAvail}

procedure HandleRequest(var commRegs : CommRegType);
  begin
  with commRegs do
    begin
    case ah of
      0: {Initialize comm port}
        begin
        Port[portBase+LCR] := Port[portBase+LCR] or DLAB; {Access divisor
                                                           latch}
        Port[portBase+DLL] := Lo(divisorTable[ah shr 5]); {Set baud rate}
        Port[portBase+DLH] := Hi(divisorTable[ah shr 5]);
        Port[portBase+LCR] := al and CHARMASK; {Set character parameters}
        ah := Port[portBase+LSR] or Byte(CharAvail);
        al := Port[portBase+MCR]; {Return modem status}
        end;
      1: {Send character}
        begin
        {Because the UART always runs, we wait at most one character time}
        repeat until (Port[portBase+LSR] and THRE) <> 0;
        Port[portBase+THR] := al;
        ah := Port[portBase+LSR] or Byte(CharAvail);
        end;
      2: {Receive character}
        begin
        if CharAvail then
          begin
          asm cli end; {Maintain consistency}
          buffOut := Succ(buffOut) and BUFFMAX;
          al := rcvBuff[buffOut];
          asm sti end; {Interrupts OK now}
          ah := (Port[portBase+LSR] or Byte(CharAvail)) and READSTATUSMASK;
          end
        else
          ah := ERRORBYTE;
        end;
      3: {Get Status}
        begin
        ah := Port[portBase+LSR] or Byte(CharAvail);
        al := Port[portBase+MCR]; {Return modem status}
        end;
      4: {Extended Initialize}
        begin
        if cl <= 8 then
          begin
          Port[portBase+LCR] := Port[portBase+LCR] or DLAB; {Access divisor
                                                             latch}
          Port[portBase+DLL] := Lo(divisorTable[cl]); {Set baud rate}
          Port[portBase+DLH] := Hi(divisorTable[cl]);
          end;
        Port[portBase+LCR] :=  ((al and 1) shr 6) {Break}
                             + ((bh and 3) shr 3) {Parity}
                             + ((bl and 1) shr 2) {Stop bits}
                             + ((ch + 1) and 3);  {Data bits}
        ah := Port[portBase+LSR] or Byte(CharAvail);
        al := Port[portBase+MCR]; {Return modem status}
        end;
      5: {Modem Control}
        begin
        if al = 1 then
          begin
          Port[portBase+MCR] := bl;
          ah := Port[portBase+LSR] or Byte(CharAvail);
          al := Port[portBase+MCR]; {Return modem status}
          end
        else
          bl := Port[portBase+MCR];
        end;
      end; {case}
    end; {with}
  end; {HandleRequest}

procedure FieldRequests;
  var
    serverRcvBlock, serverSendBlock : NCB;
    commRegs : CommRegType;
  begin
  serverRcvBlock.Init(RECEIVE);
  serverSendBlock.Init(SEND);
  while TRUE do
    begin
    with serverRcvBlock do
      begin
      len := SizeOf(commRegs);
      bufPtr := Addr(commRegs);
      lsn := commSessionNum;
      case serverRcvBlock.ReturnCode of
        TIMEOUT: if UserAbort then
          begin
          if NetToolsHangup(commSessionNum) <> GOOD_RTN then;
          Shutdown
          end;
        ILL_SESSION,SESSION_ABEND: {These mean session over}
          begin
          Writeln('Session aborted');
          Exit;
          end;
        SESSION_CLOSED:
          begin
          Writeln('Client closed session');
          Exit;
          end;
        GOOD_RTN:
          begin
          if len <> SizeOf(commRegs) then {Kill session if bad packet}
            begin
            if NetToolsHangup(commSessionNum) <> GOOD_RTN then;
            Exit
            end;
          HandleRequest(commRegs);
          with serverSendBlock do
            begin
            bufPtr := Addr(commRegs);
            len := SizeOf(commRegs);
            lsn := commSessionNum;
            if ReturnCode <> GOOD_RTN then
              Exit; {Session dead if send didn't complete}
            end
          end
        end {case}
      end {with}
    end {while}
  end; {FieldRequests}

begin {CommServer}
CheckBreak := FALSE;
DirectVideo := TRUE;
Writeln('CommServer V1.00, Copyright 1991 by L. Brett Glass');
ValidateParms;
if not NetBIOSPresent then
  begin
  Writeln('Error: NetBIOS not present');
  Halt
  end;
if NetToolsAddUniqueName(portName,portNameNum) <> GOOD_RTN then
  begin
  Writeln('Error: Could not register port name');
  Halt
  end;
InitPort;
while TRUE do
  begin
  Writeln('Awaiting a client...');
  AwaitClient;
  Writeln('Connection established with client ', HexStr(clientName,16));
  FieldRequests;
  Writeln('End of session');
  end;
end.
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Spying on NetBIOS

Assembly language routines to access NetBIOS datagrams and a snooping utility



By John Otken

NetBIOS—a network interface for DOS programs—permits two or more computers to exchange data quickly and reliably. A common use for NetBIOS is to transfer files among machines. Because NetBIOS is a software (not a hardware) specification, the same program should work over almost any kind of network. NetBIOS drivers exist for Ethernet, Arcnet, and Token Ring.

NetBIOS has two types of data transfer services: datagrams and sessions. *Datagrams* are useful for sending small amounts of information when it's not important to control or confirm its receipt. Some networks use datagrams to announce the time, which can help keep computer clocks synchronized. Datagrams are like postcards: you don't know whether or not the addressee receives them, and they don't necessarily arrive in the order they were sent.

A network *session* is similar to a connection between two error-correcting modems. Data can be reliably transferred in either direction, and you will know whether the connection is successful. For operations such as file transfers, sessions are preferable to datagrams.

John Otken is president of Soft Advances, where he hacks assembly language into debuggers and develops other software for embedded systems. He can be contacted by writing to him at P.O. Box 184, Orinda, CA 94563.

PLAYING THE FIELDS

Except for data buffers, all program communication with the NetBIOS driver passes through a 64-byte data structure known as the NetBIOS Control Block (NCB). The NCB has 14 fields (Figure 1). Most NetBIOS functions use only a few fields on entry to the driver, and a few more on exit. Descriptions of the fields follow.

The **Ncb_Command** byte holds the 7-bit NetBIOS function code. The most significant bit of this location is the wait option. When the wait bit is 0, the driver completes the NetBIOS function before returning to the program. When the wait bit is 1, the driver returns to the program before the function completes: the NCB is still active, and the driver updates the other fields of the NCB as the network function executes. More than one NCB can be active simultaneously.

The **Ncb_RetCode** byte holds the final return code for a NetBIOS function. While a network function is in progress, this field is 0xFF. After a function completes, the driver sets the return code in both **Ncb_RetCode** and **Ncb_Cmd_Cplt.** Programs should use **Ncb_Cmd_Cplt** (not **Ncb_RetCode**) to detect when a function has completed. Table 1 (p. 51) lists some common return codes.

Ncb_Lsn is a 1-byte local session number. The NetBIOS **call** and **listen** functions exit with this value. Because more than one session can be active, the **send, receive,** and **hang up** functions use **Ncb_Lsn** to identify the intended session.

The **Ncb_Num** byte associates a local name number with the datagram functions. The NetBIOS **add name** function returns this number. Programs must ensure that this field is valid before using any datagram functions. The session functions do not use this field; they use **Ncb_Name** to identify a local process.

Ncb_BufferOff and **Ncb_BufferSeg** contain a 32-bit pointer to a buffer for data transfers. **Ncb_Length** is the length of that buffer. If a buffer is too small for a **receive** function, the driver sets a return code of 0x06.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

LISTING 1

```
   ***NOTE: minor edits have been made in order to conserve space.-Ed.***

; netbios.inc

;
; NetBIOS command codes
;
NB_CALL_WAIT                    equ  10h
NB_CALL                         equ  90h
NB_LISTEN_WAIT                  equ  11h
NB_LISTEN                       equ  91h
NB_HANG_UP_WAIT                 equ  12h
NB_HANG_UP                      equ  92h
NB_SEND_WAIT                    equ  14h
NB_SEND                         equ  94h
NB_RECEIVE_WAIT                 equ  15h
NB_RECEIVE                      equ  95h
NB_SEND_DATAGRAM_WAIT           equ  20h
NB_SEND_DATAGRAM                equ 0A0h
NB_RECEIVE_DATAGRAM_WAIT        equ  21h
NB_RECEIVE_DATAGRAM             equ 0A1h
NB_SEND_BROADCAST_WAIT          equ  22h
NB_SEND_BROADCAST               equ 0A2h
NB_RECEIVE_BROADCAST_WAIT       equ  23h
NB_RECEIVE_BROADCAST            equ 0A3h
NB_ADD_NAME_WAIT                equ  30h
NB_ADD_NAME                     equ 0B0h
NB_DELETE_NAME_WAIT             equ  31h
NB_DELETE_NAME                  equ 0B1h
NB_RESET                        equ  32h
NB_CANCEL                       equ  35h
NB_ADD_GROUP_NAME_WAIT          equ  36h
NB_ADD_GROUP_NAME               equ 0B6h

NB_INVALID_COMMAND              equ  7Fh

DATAGRAM_MAX            equ     512     ; maximum length of a datagram
OK_RetCode              equ     0

;
; NetBIOS Command Block
;
ncb     struc
  Ncb_Command           db      ?       ; command (function) code
  Ncb_RetCode           db      ?       ; return code or 0xFF if pending
  Ncb_Lsn               db      ?       ; local session number
  Ncb_Num               db      ?       ; name number returned by AddName
  Ncb_BufferOff         dw      ?       ; message buffer offset
  Ncb_BufferSeg         dw      ?       ; message buffer segment
  Ncb_Length            dw      ?       ; message buffer length
  Ncb_CallName          db  16 dup(?)   ; remote name
  Ncb_Name              db  16 dup(?)   ; local name (see AddName)
  Ncb_Rto               db      ?       ; receive timeout
  Ncb_Sto               db      ?       ; send timeout
  Ncb_PostOff           dw      ?       ; pointer to procedure to execute
  Ncb_PostSeg           dw      ?       ; when nowait function completes
  Ncb_Lana_Num          db      ?       ; 0 primary adapter, 1 secondary
  Ncb_Cmd_Cplt          db      ?       ; 0 when nowait function completes
  Ncb_Reserved          db  14 dup(?)   ; reserved for use by NetBIOS
ncb     ends
; member names from "C Programmer's Guide to NetBIOS" by WDS

                                        ; end of netbios.inc
```

LISTING 2

```
; net_misc.asm
        title   NetBIOS Miscellaneous
        include asm.inc
        include netbios.inc

        public  clear_ncb, public  netbios_add_name
        public  netbios_call, public  netbios_cancel
        public  netbios_check, public  netbios_delete_name
        public  netbios_hang_up, public  netbios_receive_wait
        public  netbios_send_broadcast, netbios_send_datagram
        public  netbios_send_wait, public  put_netbios_name

        .data?

temp_ncb ncb    <>      ; netbios control block for temporary use

        .code

        extn    putchar,strncpy,save_most

;;      clear ncb
;
;       entry   DS:SI   ncb ptr
;
clear_ncb proc
        pushm   ax,cx,di,es
        mov     ax,ds
        mov     es,ax
        mov     di,si
        xor     ax,ax
        mov     cx,size ncb
        rep     stosb
        popm    es,di,cx,ax
        ret
clear_ncb endp

;;      netbios add name
;
;       entry   DS:SI   asciiz name (less than 16 characters long)
;       exit    Cf      if error
;               AL      name number (or return code if error)
netbios_add_name proc
        call    save_most
        call    zero_temp_ncb           ; copy name to control block
        call    write_temp_ncb
        lea     di,Ncb_Name[di]
        mov     cx,size Ncb_Name+1      ;  (+1 is for \0, OK to overflow \0
        call    strncpy                 ;   into next field)

        call    read_temp_ncb
        mov     Ncb_Command[si],NB_ADD_NAME_WAIT
        call    netbios_call
        jc      nan1                    ;  if error

        mov     al,Ncb_Num[si]
nan1:   ret
netbios_add_name endp

;;      netbios call
;
;       entry   DS:SI   netbios control block
;       exit    AX      return code (0xFF if function pending)
;               Cf      if error
; netbios_call proc
        push    es
        mov     ax,ds                   ; call NetBIOS
        mov     es,ax                   ;  ES:BX points to control block
        xchg    bx,si
        int     5Ch
        xchg    bx,si

        mov     al,Ncb_Cmd_Cplt[si]     ; get return code
        and     ax,0FFh                 ;  (AH=0)
        jz      ncb1                    ;  if successful completion (Cf==0)

        cmp     al,0FFh                 ; set Cf if error (AL!=0xFF)

ncb1:   pop     es
        ret
netbios_call endp

;;      netbios cancel
;
;       entry   DS:SI   netbios command block
;       exit    AX      final return code
;               Cf      if error
netbios_cancel proc
        pushm   si,ds
        pushm   si,ds                   ; (push again, pop into NCB below)

        call    read_temp_ncb           ; clear control block
        call    clear_ncb

        pop     Ncb_BufferSeg[si]       ; set control block fields
        pop     Ncb_BufferOff[si]
        mov     Ncb_Command[si],NB_CANCEL

        call    netbios_call
        popm    ds,si
        ret
netbios_cancel endp

;;      netbios check
```

continued

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The 16-byte **Ncb_CallName** field holds the name of the remote process for sessions and datagrams. A program sets this field on entry to **call, listen,** or **send** datagram functions. Both **receive** datagram functions set this field. A program can use the special name "*" to listen for a call from any process.

The 16-byte **Ncb_Name** field holds the local process name for the **call, listen, add name,** and **delete name** functions. The datagram functions do not use this field; they use **Ncb_Num.** Programs must register names using the **add name** function before a call or a listen.

The **Ncb_Rto** and **Ncb_Sto** bytes hold the session **receive** and **send** timeout values in half-second intervals. A value of 0 disables timeouts. These fields are valid on **call** and **listen** functions only.

Ncb_PostOff and **Ncb_PostSeg** contain a pointer to a postprocessing procedure. When a function completes, the driver invokes the routine at this address. On entry, register AL has the final return code, and ES:BX points to the NCB. **Ncb_Cmd_Cplt** is still 0xFF. The procedure should process the function and return to the driver with an IRET; it should not make any calls to DOS, and it does not have to save registers. The NetBIOS driver ignores this field if it is 0 or if the wait bit of **Ncb_Command** is 0 (wait-enabled).

On systems that have multiple network adapter cards, **Ncb_Lana_Num** selects the card to associate with an NCB. On most systems, this field is 0.

The **Ncb_Cmd_Cplt** field holds the final return code for NetBIOS functions. It is identical to **Ncb_RetCode** except that the driver does not set this field until after it has executed the **Ncb_Post** procedure. Programs should use this field to detect when a function completes.

The 14-byte **Ncb_Reserved** field is reserved for the NetBIOS driver. Programs should not manipulate this field.

FIGURE 1: The NetBIOS control block

```
Ncb_Command     db      ?           ; function code
Ncb_RetCode     db      ?           ; return code
Ncb_Lsn         db      ?           ; local session number
Ncb_Num         db      ?           ; local name number
Ncb_BufferOff   dw      ?           ; data buffer offset
Ncb_BufferSeg   dw      ?           ; data buffer segment
Ncb_Length      dw      ?           ; data buffer length
Ncb_CallName    db 16 dup(?)        ; remote process name
Ncb_Name        db 16 dup(?)        ; local process name
Ncb_Rto         db      ?           ; receiver time out
Ncb_Sto         db      ?           ; sender time out
Ncb_PostOff     dw      ?           ; post proc offset
Ncb_PostSeg     dw      ?           ; post proc segment
Ncb_Lana_Num    db      ?           ; adapter number
                                    ;  (usually 0)
Ncb_Cmd_Cplt    db      ?           ; command complete
                                    ;  flag
Ncb_Reserved    db 14 dup(?)        ; reserved for driver
```

LOOKING FOR MR. NETBIOS

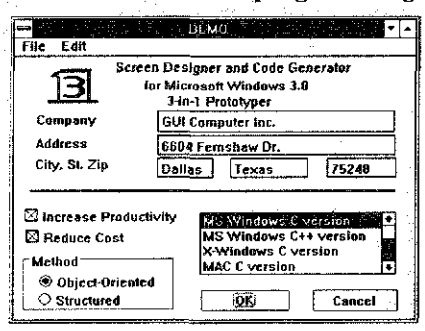
NetBIOS is an optional part of a computer's system software. Before executing any functions, a program should check for the presence of a NetBIOS driver. Sample code for this

C/C++ Windows 3.0

Screen Designer Code Generator

Only $159

The easiest way to learn and to master Microsoft Windows programming



With 3-in-1, developing this window screen requires only

20 Minutes !!

That includes screen layout, code generation, compiling and linking.

GUI's 3-in-1 C $159 C++ $199 Both $259

30 Day Guarantee No Source Code Royalties

Also many low cost Windows source code utilities.

1-800-800-9010 TEL 214-250-3472 Fax 214-250-1355

Circle 39 on Reader Service Card

If your Windows application needs images, you need ImageMan!

ImageMan provides a complete library for displaying and printing virtually any type of image. Our object-oriented approach allows your application to access all images in the same manner -- regardless of the type of image, your code doesn't change!

- Saves months of development effort.
- Use with any Windows 3.0 environment that supports DLLs, including: C, Pascal, Actor, Toolbook, Plus, etc...
- Completely supports TIFF, PCX, EPSF, WMF, and BMP images (with more to come....)
- Easily extended to support new image formats.
- Uses assembly language routines for *Speed.*
- Highly configurable - you have *full* control.
- Royalty Free!

Call today for free demo!

DTI Data Techniques, Inc.
1000 Business Center Drive, #120
Savannah, GA 31405

$395
$995 w/Source

800-868-8003
912-651-8003

Circle 40 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

operation appears in the procedure **netbios_check.** Because old IBM PCs zero unused interrupt vectors, this routine first ensures that the vector for interrupt 0x5C is nonzero. Then **netbios_check** will execute an illegal NetBIOS function, and if the function returns an error, then a NetBIOS driver is present.

A program controls a network by building a NetBIOS control block and invoking the driver through interrupt 0x5C. On entry to the driver, registers ES:BX point to the NCB. The driver exits with a return code in AL, leaving the other registers unchanged. The only field required for all NetBIOS functions is a code in **Ncb_Command.**

NETBIOS FUNCTIONS

NetBIOS has 21 different functions; 18 of them accept the wait option, and the other 3 always wait for a function to complete.

The sample programs use 12 functions, described below. Table 2 summarizes the entry and exit conditions for these functions.

```
;
;       exit    Cf      if no netbios driver
;       uses    AX,SI
netbios_check proc
        push    ds                      ; check if vector 5C is zero
        mov     ax,0
        mov     ds,ax
        mov     si,4*5Ch
        lds     si,[si]                 ;  (this faults in protect mode)
        mov     ax,ds
        or      ax,si
        jz      nbcl                    ;  if no NetBIOS driver (Cf==0)

        call    read_temp_ncb
        call    clear_ncb
        mov     Ncb_Command[si],NB_INVALID_COMMAND
        call    netbios_call

nbcl:   cmc                             ; flip error flag
        pop     ds
        ret
netbios_check endp

;;      netbios delete name
;
;       entry   DS:SI   asciiz name (less than 16 characters long)
;       exit    Cf      if error
;               AL      return code
netbios_delete_name proc
        call    save_most
        call    zero_temp_ncb           ; copy name to control block
        call    write_temp_ncb
        lea     di,Ncb_Name[di]
        mov     cx,size Ncb_Name+1      ;  (+1 is for \0, OK to overflow
\0      call    strncpy                 ;   into next field)

        call    read_temp_ncb
        mov     Ncb_Command[si],NB_DELETE_NAME_WAIT
        call    netbios_call
        ret
netbios_delete_name endp

;;      netbios hang up
;
;       entry   AL      session number
;       exit    AL      return code
;               Cf      if error
netbios_hang_up proc
        pushm   si,ds
        call    read_temp_ncb
        call    zero_temp_ncb
        mov     Ncb_Lsn[si],al
        mov     Ncb_Command[si],NB_HANG_UP_WAIT
        call    netbios_call
        popm    ds,si
        ret
netbios_hang_up endp

;;      netbios receive wait
;
;       entry   AL      local session number
;               CX      buffer length
;               ES:DI   buffer pointer
;       exit    AL      return code
;               CX      bytes received (0 if session closed or error)
;               Cf      if error (session closed is not an error)
netbios_receive_wait proc
        pushm   si,ds
        call    read_temp_ncb
        call    zero_temp_ncb
        mov     Ncb_Lsn[si],al
        mov     Ncb_Length[si],cx
        mov     Ncb_BufferOff[si],di
        mov     Ncb_BufferSeg[si],es

        mov     Ncb_Command[si],NB_RECEIVE_WAIT

        call    netbios_call
        jnc     nrw1                    ; if receive OK

        mov     cx,0                    ; else check return code
        cmp     al,0Ah
        je      nrw2                    ;  if session closed, then no error
        stc                             ;  else return error
        jmp     nrw2

nrw1:   mov     cx,Ncb_Length[si]
nrw2:   popm    ds,si
        ret
netbios_receive_wait endp

;;      netbios send broadcast
;
;       entry   AL      add name number
;               CX      buffer length
;               ES:DI   buffer
;       exit    AL      return code
;               Cf      if error
;       note    waits until datagram sent over network
netbios_send_broadcast proc
        pushm   si,ds
        call    read_temp_ncb
        call    clear_ncb
        mov     Ncb_Num[si],al
        mov     Ncb_Length[si],cx
        mov     Ncb_BufferOff[si],di
        mov     Ncb_BufferSeg[si],es
        mov     Ncb_Command[si],NB_SEND_BROADCAST_WAIT
        call    netbios_call
        popm    ds,si
        ret
netbios_send_broadcast endp

;;      netbios send datagram
;
;       entry   AL      add name number
;               CX      buffer length
;               ES:DI   buffer
;               DS:SI   asciiz call name (destination)
;       exit    AL      return code
;               Cf      if error
;       note    waits until datagram sent over network
netbios_send_datagram proc
        call    save_most               ; initialize datagram control block
        pushm   ds,si
        call    zero_temp_ncb
        call    read_temp_ncb
        mov     Ncb_Num[si],al          ;  set "add name" number,
        mov     Ncb_Length[si],cx       ;   buffer length and pointer
        mov     Ncb_BufferOff[si],di
        mov     Ncb_BufferSeg[si],es
        popm    si,ds

        call    write_temp_ncb          ; copy destination name into block
        lea     di,Ncb_CallName[di]
        mov     cx,size Ncb_CallName+1
        call    strncpy

        call    read_temp_ncb           ; send datagram and wait
        mov     Ncb_Command[si],NB_SEND_DATAGRAM_WAIT
```

continued

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
        call    netbios_call
        ret
netbios_send_datagram endp

;;      netbios send wait
;
;       entry   AL      session number
;               CX      length
;               ES:DI   buffer pointer
;       exit    AL      return code
;               Cf      if error
netbios_send_wait proc
        pushm   si,ds
        call    read_temp_ncb
        call    zero_temp_ncb

        mov     Ncb_Lsn[si],al
        mov     Ncb_Length[si],cx
        mov     Ncb_BufferOff[si],di
        mov     Ncb_BufferSeg[si],es

        mov     Ncb_Command[si],NB_SEND_WAIT
        call    netbios_call
        popm    ds,si
        ret
netbios_send_wait endp

;;      put netbios name
;
;       entry   DS:SI   name field of control block
;       uses    AX,SI
put_netbios_name proc
        push    cx
        mov     cx,size Ncb_CallName
pnn1:   lodsb
        cmp     al,' '
        jbe     pnn2
        cmp     al,'~'
        jbe     pnn3
pnn2:   mov     al,' '
pnn3:   call    putchar
        loop    pnn1
        pop     cx
        ret
put_netbios_name endp

;;      read temp ncb
;
;       exit    DS:SI   temporary netbios control block
read_temp_ncb proc
        mov     si,@data
        mov     ds,si
        lea     si,temp_ncb
        ret
read_temp_ncb endp

;;      write temp ncb  ;
;       exit    ES:DI   temporary netbios control block
write_temp_ncb proc
        mov     di,@data
        mov     es,di
        lea     di,temp_ncb
        ret
write_temp_ncb endp

;;      zero temp ncb
zero_temp_ncb proc
        pushm   ax,si,ds
        call    read_temp_ncb
        call    clear_ncb
        popm    ds,si,ax
        ret
zero_temp_ncb endp
        end
```

LISTING 3

```
; datagram.asm
        title   send datagram
        include asm.inc
        include netbios.inc

        public  main

        .stack

        .const
add_name_text   db      'Send Datagram',0

broadcast_text  db      'Broadcast '
datagram_text   db      'Datagram sent.',0

help_text       label   byte
  db  'datagram <target> <text>',13,10
  db  'Sends <text> in NetBIOS datagram to <target>.',13,10
  db  'Use * to broadcast datagram.',13,10
  db  'e.g.: datagram TEST This is a datagram to remote name TEST.',13,10
  db  'e.g.: datagram * This is a broadcast datagram.',0

driver_ertx     db      'No NetBIOS driver found',0
add_name_ertx   db      'Add Name failed',0

        .data?
        extrn   argc:word,argv:dword

add_name db     ?

        .code
        extn    startup,puts,exit_program,netbios_check,netbios_add_name
        extn    strskp_white,read_command_line,netbios_send_datagram
        extn    netbios_send_broadcast,netbios_delete_name

;;      help
;
;       entry   DS      DGROUP
help    proc
        lea     si,help_text
        call    puts
        jmp     exit_program
help    endp

;;      main
;
main    proc
        cmp     argc[bp],2
        jb      help                    ;  if no arguments

        call    netbios_check
        jc      mai5                    ;  if no netbios driver

        lea     si,add_name_text        ; *Send Datagram*
        call    netbios_add_name
        jc      mai6                    ;  if add name failed
        mov     add_name[bp],al

        call    read_command_line       ; find <text> in command line
        call    strskp_white
mai1:   lodsb                           ;  skip <target>
        cmp     al,' '
        ja      mai1

        dec     si                      ;  skip space between <target>&<text>
        call    strskp_white

        mov     di,si                   ;  ES:DI points to <text>
        mov     ax,ds
        mov     es,ax
        mov     cx,-1                   ; compute length of <text>
mai2:   inc     cx
        lodsb
        cmp     al,' '                  ;  (any control char is delimiter)
        jae     mai2                    ;  if delimiter

        mov     al,add_name[bp]

        lds     si,argv[bp]
        mov     si,[si+2]
        cmp     word ptr [si],'*'
        je      mai3                    ;  if broadcast datagram

        call    netbios_send_datagram   ;  else send normal datagram
        lea     si,datagram_text
        jmp     mai4

mai3:   call    netbios_send_broadcast
        lea     si,broadcast_text
mai4:   mov     ax,@data
        mov     ds,ax
        call    puts
        lea     si,add_name_text
```

continued

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
        call    netbios_delete_name
        jmp     exit_program

mai5:   lea     si,driver_ertx          ; *No NetBIOS driver found*
        call    puts
        jmp     exit_program

mai6:   lea     si,add_name_ertx        ; *Add Name failed*
        call    puts
        jmp     exit_program
main    endp

        end
```

LISTING 4

```
; spy.asm
        title   netbios datagram spy
        include asm.inc
        include netbios.inc

        public  main

        .stack

        .const
broadcast_text  db  'Broadcast',0
callname_text   db  ' - Call Name:',0
datagram_text   db  'Datagram ',0
driver_text     db  'No NetBIOS driver found',0
help_text       db  'NetBIOS Spy - displays network datagrams',13,10
                db  'Press ESC to exit',0
length_text     db  ' - Length:',0
retcode_text    db  ' - RetCode:',0
spy_text        db  'Spy',0

        .data?
ncb_broadcast   ncb <>                  ; control blk for broadcast
datagrams
ncb_datagram    ncb <>                  ; control blk for other datagrams

buf_broadcast   db  DATAGRAM_MAX dup(?) ; buffer for broadcast datagrams
buf_datagram    db  DATAGRAM_MAX dup(?) ; buffer for other datagrams

add_name db     ?

        .code
 extrn exit_program,get_input_state,clear_ncb,netbios_call,netbios_cancel
 extrn putchar_newline,puts,put_hex_byte,put_hex_word,put_multiple_bytes
 extrn put_netbios_name,put_string,startup,netbios_check
 extrn netbios_add_name,netbios_delete_name

;;      main
;
main    proc
        lea     si,help_text
        call    puts

        call    netbios_check
        jc      mai4                    ;  if no netbios driver

        lea     si,spy_text
        call    netbios_add_name
        mov     add_name[bp],al
mai1:   lea     si,ncb_broadcast        ; check receive broadcast datagram
        cmp     Ncb_Cmd_Cplt[si],0FFh
        je      mai2                    ;  if receive pending

        call    clear_ncb               ;  else post receive broadcast
        mov     Ncb_Command[si],NB_RECEIVE_BROADCAST
        mov     Ncb_Num[si],0FFh        ;  (FF means any datagram)
        mov     Ncb_BufferOff[si],offset buf_broadcast
        mov     Ncb_BufferSeg[si],ds
        mov     Ncb_Length[si],size buf_broadcast
        call    netbios_call

mai2:   lea     si,ncb_datagram         ; check receive datagram
        cmp     Ncb_Cmd_Cplt[si],0FFh
        je      mai3                    ;  if receive pending

        call    clear_ncb               ;  else post receive broadcast
        mov     Ncb_Command[si],NB_RECEIVE_DATAGRAM
        mov     al,0FFh                 ;  (FF means any datagram)
        mov     al,add_name[bp]
        mov     Ncb_Num[si],al
        mov     Ncb_BufferOff[si],offset buf_datagram
        mov     Ncb_BufferSeg[si],ds
        mov     Ncb_Length[si],size buf_datagram
        call    netbios_call

mai3:   call    get_input_state
        jnz     mai5                    ;  if key waiting

        lea     si,ncb_broadcast
        call    process_control_block

        lea     si,ncb_datagram
        call    process_control_block
        jmp     mai1

mai4:   lea     si,driver_text          ; *No NetBIOS driver found*
        call    puts
        jmp     exit_program
mai5:   lea     si,ncb_broadcast
        call    netbios_cancel
        lea     si,ncb_datagram
        call    netbios_cancel
        lea     si,spy_text
        call    netbios_delete_name
        jmp     exit_program
main    endp

;;      process control block
;
;       entry   DS:SI   datagram control block
;       uses    AX,BX,CX,SI
process_control_block proc
        mov     bx,si
        cmp     Ncb_Cmd_Cplt[bx],0FFh
        je      pcb3                    ;  if receive still pending
        lea     si,datagram_text        ; *Datagram *
        cmp     Ncb_Command[bx],NB_RECEIVE_DATAGRAM
        je      pcb1
        lea     si,broadcast_text       ; *Broadcast*
pcb1:   call    put_string

        cmp     Ncb_RetCode[bx],0       ; display non-zero return code
        je      pcb2

        lea     si,retcode_text         ; * - RetCode:*
        call    put_string
        mov     al,Ncb_RetCode[bx]
        call    put_hex_byte

pcb2:   lea     si,length_text          ; * - Length:*
        call    put_string
        mov     ax,Ncb_Length[bx]
        call    put_hex_word

        lea     si,callname_text        ; * - Call Name:*
        call    put_string

        lea     si,Ncb_CallName[bx]     ; display call name
        call    put_netbios_name
        call    putchar_newline

        push    ds                      ; display datagram
        mov     cx,Ncb_Length[bx]
        lds     si,dword ptr Ncb_BufferOff[bx]
        call    put_multiple_bytes
        pop     ds
pcb3:   ret
process_control_block endp

        end
```

LISTING 5

```
; server.asm
        title   simple file server
        include asm.inc
        include netbios.inc

        public  main

        .stack

        .const
ertx_add_name           db  'Add Name error',0
ertx_driver             db  'No NetBIOS driver',0
```

continued

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The ADD NAME function registers a name into the NetBIOS local name table and returns a local name number. The name must be unique across the network. To register a name, a program fills the name into the **Ncb_Name** field, sets the **Ncb_Command** code, and calls the NetBIOS driver. Names are always 16 bytes long. NetBIOS does not use null terminating bytes (as in C), nor does it use string lengths (as in Pascal).

The driver returns a numeric identifier for the name in the **Ncb_Num** field; use this number with the various datagram functions. Because this function must wait for possible replies from other systems on the network, it can take several seconds to complete. Programs should delete names before they exit.

The CALL function opens a full duplex session with another process on the network. The entry conditions for CALL require a target name in the **Ncb_CallName** field, a registered local name in **Ncb_Name,** and timeout values in **Ncb_Rto** and **Ncb_Sto.** If another process has a LISTEN posted with the matching names, the CALL function returns a local session number in **Ncb_Lsn.**

The CANCEL function aborts a pending NetBIOS function. CANCEL cannot abort a waiting function, nor can it abort the ADD NAME, DELETE NAME, SEND DATAGRAM, or SEND BROADCAST DATAGRAM functions. This operation is usually used on a pending **receive** or **listen** function. On entry to CANCEL, the address of the target NCB is in **Ncb_BufferOff** and **Ncb_BufferSeg.**

DELETE NAME removes a 16-character name from a NetBIOS local name table. Before deleting a name, a program should cancel functions or terminate sessions that use the name. If a session is active, NetBIOS can defer a **delete** operation, but error detection and recovery become more complex.

HANG UP closes the session identified by the local session number in **Ncb_Lsn.** Either side can close a session, but it is usually easier to let the sending side do so. A HANG UP at either end closes the session for both ends. When a session closes, a pending **send** function completes, but a pending **receive** function returns 0x0A in its **Ncb_RetCode** field.

The LISTEN function opens a session with a corresponding CALL function. Entry conditions for LISTEN require a target name in the **Ncb_CallName** field, a registered local name in **Ncb_Name,** and timeout values in **Ncb_Rto** and **Ncb_Sto.** NetBIOS establishes a session between the CALL and LISTEN functions when the **Ncb_Name** fields of both sides match the **Ncb_CallName** fields of the other. The LISTEN function can use the special wildcard name "*" to match any calling **Ncb_Name** field. As with CALL, the **Ncb_Name** field must be registered with ADD NAME. When a session opens, NetBIOS puts a local session number in **Ncb_Lsn.**

The RECEIVE function accepts data from a session partner. Entry conditions require a local session number in **Ncb_Lsn** and a buffer specification in **Ncb_BufferOff, Ncb_BufferSeg,** and **Ncb_Length.** If the receiving buffer is too small, NetBIOS fills the buffer and returns a code of 0x06. The program must post another RECEIVE function to transfer the rest of the data. If a session closes, active RECEIVE functions set **Ncb_RetCode** to 0x0A. On exit, **Ncb_Length** holds the number of bytes transferred.

RECEIVE BROADCAST DATAGRAM accepts datagrams from any process that issues a SEND BROADCAST DATAGRAM. Entry conditions require a valid local name number in **Ncb_Num** and a buffer specification in **Ncb_BufferOff, Ncb_BufferSeg,** and **Ncb_Length.** If the buffer is too small, the function code is 0x06, and the remaining data is lost. On exit, **Ncb_Length** holds the number of bytes that were sent.

RECEIVE DATAGRAM accepts a datagram message from a matching SEND DATAGRAM. The sending **Ncb_CallName** field must match the name that corresponds to the **Ncb_Num** field of the receiver. The receiver can use the wildcard value 0xFF to match any locally registered name. The key word here is "registered"; a program is not able to receive all possible datagrams because it cannot register all possible names. As with all **receive** functions, the program must specify a buffer.

The SEND function transfers data to

TABLE 1: Common NetBIOS return codes

00h	Function complete, no errors
01h	Invalid buffer length
03h	Invalid command
05h	Function timeout
06h	Incomplete receive message (buffer too small)
08h	Bad local session number
0Ah	Session closed
0Bh	Command canceled
0Dh	Duplicate name in local name table
0Eh	Name table full
0Fh	Name deleted with sessions active
11h	Local session table full
15h	Name not found or invalid
16h	Name in use on remote
18h	Session ended abnormally
19h	Name conflict detected
FFh	Command pending

TABLE 2: Entry and exit conditions for some NetBIOS functions

Function	Command Code	Return Code	Local Sessn	Name Num	Mssg Buffer	Buffer Length	Remote Name	Call Name	-Timeouts- Recv	Send
ADD NAME	30h/B0h	Out		Out			In			
CALL	10h/90h	Out	Out				In	In	In	In
CANCEL	35h	Out			In					
DELETE NAME	31h/B1h	Out						In		
HANG UP	12h/92h	Out	In							
LISTEN	11h/91h	Out	Out				In*	In	In	In
RECEIVE	15h/95h	Out	In		In	In/Out				
RECEIVE BROADCAST DATAGRAM	23h/A3h	Out		In**	In	In/Out	Out			
RECEIVE DATAGRAM	21h/A1h	Out		In	In	In/Out	Out			
SEND	14h/94h	Out	In		In	In				
SEND BROADCAST DATAGRAM	22h/A2h	Out		In	In	In				
SEND DATAGRAM	20h/A0h	Out		In	In	In	In			

* LISTEN can use a CallName of "*" to match any name field of any CALL.
**RECEIVE DATAGRAM can set the Ncb_Num field to 0xFF to match any register local name.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Debug C

What's *your* problem?

- *program hanging?*
- *memory overwrites?*
- *operating system overwrites?*
- *corrupted heap?*
- *memory leaks?*
- *dangling pointers?*
- *open file pointers?*
- *invalid calls to free()?*

If your code has a bug that's making *you* sick, *call the doctor*—**Dr. MD, the run time memory debugger for C!** Sure, you *could* find it yourself, but how much would it cost you in time, money and aggravation?

Dr. MD is the **easy and cost effective** way to cure your memory corruption ills. **Dr. MD** gives a **complete diagnosis** including the source file and line number of your bug.

- *includes source code*
- *non obtrusive*
- *easy to customize*
- *hardware independent (386 not required)*
- *operating system independent (DOS, UNIX, OS/2...)*
- *small library*

Start your code on the road to recovery—order your copy today!

DR. MD™

NEW version 2.0
$129.00 plus shipping

RSS Engineering
Custom and Contract
Software Solutions

2683 N. Taft
Loveland, CO 80538 USA
(303) 669-3300
FAX (303) 669-3354

MasterCard VISA

RSS Engineering
a division of
Remote Switch Systems, Inc.

Circle 41 on Reader Service Card

a session partner. This function is entered with a local session number in **Ncb_Lsn** and a buffer specification. Note that the maximum buffer length is 65,535. On exit, SEND returns values in **Ncb_RetCode** and **Ncb_Cmd_Cplt.** Sessions guarantee data integrity and order: if a SEND does not complete correctly, the session aborts.

SEND BROADCAST DATAGRAM transmits a message to every network host. Every process with a pending **receive broadcast** function receives the message.

SEND DATAGRAM transmits a message to the network. A process receives the message if it has a pending RECEIVE DATAGRAM having an **Ncb_Num** field that matches the **Ncb_CallName** of the sender. Only one process per system can receive the datagram. SEND DATAGRAM requires a buffer specification, a local name number in **Ncb_Num,** and a remote name in **Ncb_CallName.**

SENDING DATAGRAMS

DATAGRAM.ASM (Listing 3) is a simple program to send datagrams. It uses the first command-line argument for the remote name. If the argument is "*," it sends a broadcast datagram; the remainder of the command line forms the datagram message. This is as simple as a NetBIOS program can get.

The program follows six steps:

- Check for a NetBIOS driver.
- Register name in local name table.
- Parse the command line.
- Build either a normal or a broadcast datagram NCB.
- Send the datagram.
- Delete its name from the name table.

SPY VS. SPY

SPY.ASM (Listing 4) is a program to receive broadcast datagrams and datagrams sent to "Spy." It displays the message contents in both hex and ASCII. The program is supposed to display all NetBIOS datagrams, but NetBIOS ignores datagrams sent to unregistered names.

The main loop in SPY.ASM issues two NetBIOS functions: a **receive** datagram and a **receive broadcast** datagram. Both functions set the no-wait option, so NetBIOS immediately returns control to the program. The program polls each NCB and the keyboard for activity. When a datagram arrives, SPY dumps it to the console and issues another **receive** function.

If two broadcast datagrams occur in rapid succession, SPY can miss the second datagram. This trade-off keeps the program simple.

Pressing a key cancels the pending **receive** functions and deletes the local name. The program then exits. Do not press Control-C to abort the program during a display operation. Aborting the program would leave an active NCB, and the system would probably crash. If you accidentally press Control-C, reset the computer.

USING SESSIONS

SERVER.ASM (Listing 5) is a simple network file server. It issues a no-wait LISTEN function and waits for network or keyboard activity. If a client opens a session, the server issues a **receive.** It expects four data objects: a file position, a maximum byte count, a direction flag, and a file name.

Assuming that the 1-byte direction flag is "r" for read, the server opens the file for input and sets the position pointer. Next, the server reads the file and sends the data to the client. The server transfers data until it reaches the end of the file or the maximum byte count. After sending the last buffer, the server closes the session with a **hang up** function and issues another LISTEN for the next client. If a keystroke occurs, the server cancels the listen, deletes its local name, and exits.

CLIENT.ASM (available on the code disk and over *PJ* bulletin boards) is a simple file transfer program. After registering its local name, it issues a CALL to a file server. When the session opens, it sends a file position, length, direction flag, and file name. Using a file position of 0 and a length of –1 ensures that the file server will transfer the entire file.

The client opens a local output file and enters a loop. The loop reads from the network and writes to the output file until the **read** function returns a code indicating that the session closed. Then the client closes the output file, deletes its local NetBIOS name, and returns to DOS.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

NETWORK WITH CARE

Because many more errors are possible in a network environment, error handling and recovery can be complicated. All four sample programs suffer from weak error processing. The goal of the programs is to demonstrate a basic network interface, not to cover every detail of the NetBIOS communications interface.

Using the terminator feature of the start-up code could ensure that the program cleans up, no matter how it terminates. For more information, see "Real-mode virtual memory" (*PJ* 9.1).

Tying the basic network functions into the error system would permit more generic error handling. It would also eliminate the error text from the sample programs. "Errors and stream I/O" (*PJ* 8.4) provides more information on error handling.

MORE INFORMATION

A good resource on NetBIOS programming is the *C Programmer's Guide to NetBIOS*, by W. David Schwaderer (Howard W. Sams, 1988). Obviously intended for C hackers, the book also contains useful information for assembly language programmers. It also includes a wonderful reference card. Many of the symbols in the sample code come from this book. Also see the article "Exploiting NetBIOS," written by Schwaderer (*PJ* 8.1).

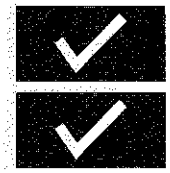
If you lack a network, the quick and easy way to experiment with NetBIOS programming is to purchase a Lantastic starter kit. The 2-megabit version is the least expensive, but the smart move is to get Western Digital Ethernet cards and run the Lantastic network operating system on top of them. If you spend some big bucks on a Novell network, you won't need new network cards. If you already have a Novell network, NetWare comes with a NetBIOS emulator. The Novell NetBIOS TSR must be invoked on your workstation after the NetWare shell is in place (IPX and NETx).

Next time, I will organize the code from past articles. I'm tired of clipping and snipping code from my library for the code disks. In future articles, each code disk will include the entire library. I will also take a look at the new version of Microsoft MASM 6.0. PJ

Some Assembly Required!

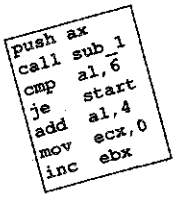
V Communications has all the tools you need to develop, examine, verify, and document assembly-language programs, quicker and easier than ever before.

New!



ASM Checker™ - Locates Coding Problems

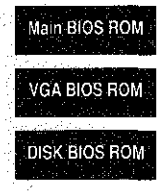
ASM Checker automatically locates common coding errors that are often hard to find and easily overlooked. It also cross-references external and public symbols and identifies unused symbols and type mismatches. It can save you countless hours of debugging time. $179.95



Sourcer 486™ - Commenting Disassembler

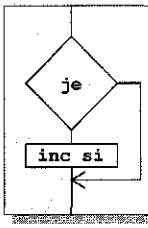
Sourcer creates commented source code and listings from memory or executable files. It helps you understand the inner details of how programs work. Sourcer output is suitable for reassembly, so you can easily make changes or correct problems in programs. $129.95

"Sourcer is the best disassembler we've ever seen!"
- PC Magazine



BIOS Pre-Processor™ - BIOS Listings

The BIOS Pre-Processor lets you use Sourcer to obtain commented listings for any BIOS ROM in your machine. Now you can understand how your specific BIOS works. Adds over 75K of comments specific to your BIOS. $49.95
Sourcer with BIOS Pre-Processor $169.95 (Save $10)



ASMtool 486™ - Flowcharting and Analysis

ASMtool makes complex assembly source easy to understand. It automatically creates flowcharts and procedure call diagrams that clarify logic paths. It also provides static timing analysis for each procedure, register usage analysis, and more. $199.95

More Performance Tools from V Communications

View-It™ - Fast, zoomable file viewer, easily view wide 132-column documents and assembly listings.	$69.95
Unpacker™ - Unpacks packed EXE files and more!	$39.95
ASM ProPak III™ - Sourcer 486 with BIOS, ASMtool 486, Unpacker, View-It, and ASM Checker. (Save $120)	$549.70
Quantasm PowerLib™ - Library of over 300 documented assembly routines with source. (Object only: $89.95)	$279.95
Quantasm Magic TSR Toolkit™ - Build your own TSRs. Documented assembly source code and examples.	$199.95
C-Clearly® - Formats C source code your way.	$129.95
Memory Commander™ - Complete 386/486 memory manager that provides up to 900K of main DOS memory.	$99.95

Specifications: Works on all PC and PC-compatible computers with DOS 2.0 or later. Assembly products support 8088 through 80486 and math instruction sets and are designed to work with Microsoft MASM or Borland TASM assemblers. View-It requires EGA/VGA/XGA for zoom features. All product names are trademarks of their respective companies.

Shipping and Handling: USA $6; Canada/Mexico $10; Other $18. CA residents add sales tax. VISA/MC/AMEX.

Call 1-800-648-8266

V Communications, Inc.

4320 Stevens Creek Blvd Suite 275 Dept. PT9 San Jose CA 95129 fax 408.296.4441 408.296.4224

Circle 42 on Reader Service Card

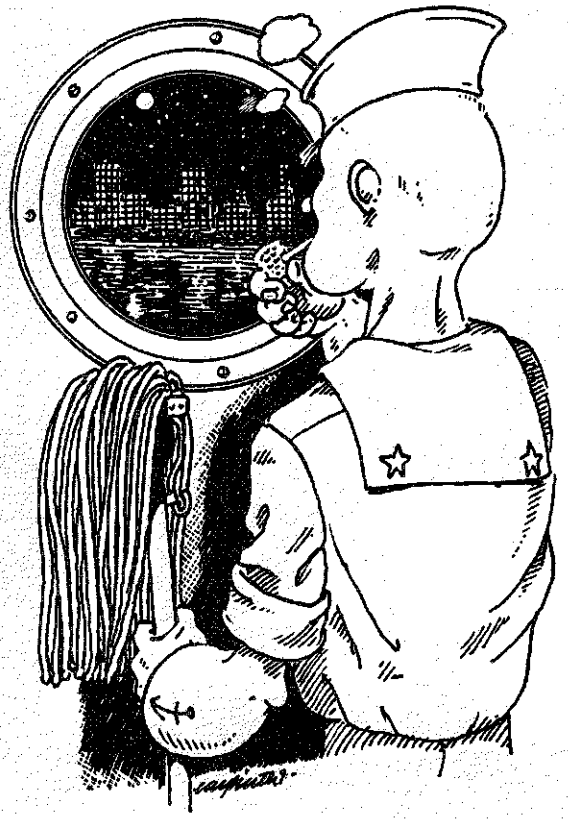
PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
ertx_send               db  'SEND error',0
ertx_listen             db  'LISTEN error',0
ertx_receive            db  'RECEIVE error',0
ertx_read_only          db  'Server only supports read requests',0
ertx_file_not_found     db  'File not found',0
ertx_read               db  'File read error',0

exiting_text            db  'Exiting',0
low_iq_server_text      db  'Low IQ Server',0
logo_text               db  'Simple NetBIOS File Server',13,10
                        db  'Press ESC to exit',0

        .data?

listen_ncb      ncb <>

;
; the order of these four objects is important
position_req    dd  ?                   ; file position requested for
                                           transfer
length_req      dd  ?                   ; length of transfer request
direction_flag  db  ?                   ; 'r' for receive, 's' for send
filename_req    db FILENAME_MAX dup(?)  ; requested file and path
                db  ?                   ; (ensure path is asciiz)

add_name db  ?
session_number  db  ?
file_buffer     db  512 dup(?)

        .code
        extn  startup,netbios_check,netbios_add_name,clear_ncb,strcpy
        extn  netbios_receive_wait,get_input_state,puts,netbios_hang_up
        extn  exit_program,netbios_delete_name,netbios_cancel,close_file
        extn  netbios_send_wait,read_from_file,select_min_count,
              netbios_call
        extn  move_file_pointer,open_input_file

;;      main
;
main    proc
        assume  ds:DGROUP               ; (test program is small model)

        lea     si,logo_text
        call    puts

        call    netbios_check
        jc      mai3                    ;  if no netbios driver
        lea     si,low_iq_server_text   ; *Low IQ Server*
        call    netbios_add_name
        jc      mai5                    ;  if add name failed
        mov     add_name,al

mai1:   lea     si,listen_ncb           ; build listen control block
        call    clear_ncb

        mov     Ncb_Command[si],NB_LISTEN
        mov     Ncb_CallName[si],'*'    ;  (accept CALLs from anyone)

        lea     si,low_iq_server_text   ;  enter server name
        lea     di,listen_ncb.Ncb_Name
        call    strcpy

        lea     si,listen_ncb
        mov     Ncb_Rto[si],-1 ;4*2     ;  receive  timeout is 4 seconds
        mov     Ncb_Sto[si],-1 ;30*2    ;  transmit timeout is 30 seconds
        call    netbios_call            ; begin listening for client
        jc      mai6                    ;  if error posting listen command

mai2:   call    get_input_state
        jnz     mai7                    ;  if keyboard exit request
        mov     al,Ncb_Cmd_Cplt[si]
        cmp     al,0FFh
        je      mai2                    ;  if listen still pending
        cmp     al,0
        jne     mai6                    ;  if listen failed

        mov     al,Ncb_Lsn[si]          ; save session number
        mov     session_number,al

        lea     di,position_req         ; get transfer request from client
        mov     cx,size position_req+size length_req+1+size filename_req
        call    netbios_receive_wait    ;  (AL is session)
        jc      mai10                   ;  if unexpected error

        cmp     direction_flag,'r'      ; check request code
        jne     mai12                   ;  if not read file

        call    process_read_request    ;  else transfer file to client
        jnc     mai1                    ;  if no errors, continue

mai3:   lea     si,ertx_driver          ; *No NetBIOS driver*
mai4:   call    puts
        jmp     exit_program

mai5:   lea     si,ertx_add_name        ; *ADD NAME error*
        jmp     mai4

mai6:   lea     si,ertx_listen          ; *LISTEN error*
        jmp     mai8

mai7:   lea     si,listen_ncb
        call    netbios_cancel
        lea     si,exiting_text         ; *Exiting*

mai8:   call    puts
mai9:   lea     si,low_iq_server_text
        call    netbios_delete_name
        jmp     exit_program

mai10:  lea     si,ertx_receive         ; *RECEIVE error*
mai11:  call    puts
        mov     al,session_number
        call    netbios_hang_up
        jmp     mai9

mai12:  lea     si,ertx_read_only    ; *Server only supports read requests*
        jmp     mai11
main    endp

;;      process read request
;
;       entry   DS      DGROUP
;       uses    AX,BX,CX,DX,SI,DI
process_read_request proc
        assume  ds:DGROUP               ; (test program is small model)

        lea     si,filename_req         ; open requested file
        call    open_input_file
        jc      prr2                    ;  if file not found

        mov     ax,wptr position_req    ; set file position
        mov     dx,wptr position_req+2
        call    move_file_pointer

prr1:   mov     cx,size file_buffer     ; choose transfer size
        mov     ax,wptr length_req
        mov     dx,wptr length_req+2
        call    select_min_count
        jcxz    prr5                    ;  if length limit reached

        lea     di,file_buffer          ; read file data
        call    read_from_file
        jc      prr3                    ;  if file read error
        mov     cx,ax
        jcxz    prr5                    ;  if end of file

        sub     wptr length_req,cx      ; update transfer limit
        sbb     wptr length_req+2,ZERO

        mov     al,session_number       ; send file data to client
        call    netbios_send_wait       ;  (ES:DI->file_buffer)
        jnc     prr1                    ;  if OK, continue transfer

        lea     si,ertx_send            ; *SEND error*
        jmp     prr4

prr2:   lea     si,ertx_file_not_found  ; *File not found*
        jmp     prr4

prr3:   lea     si,ertx_read            ; *File read error*
prr4:   call    puts
        stc
        ret
prr5:   call    close_file              ; close file and hang up session at
        mov     al,session_number       ;   end of file or when requested
        call    netbios_hang_up         ;   transfer count
        ret
process_read_request endp

        end
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Window viewports

Part 2: Adding a display layer to our text windowing system



By Allen Holub

In my last article, "Text buffers in Windows" (*PJ* 9.1), I presented the text-manipulation part of a reasonably sophisticated window-management system that compiles under Borland C++ (formerly Turbo C++), using only the C compiler. Although the code is being tested with the 2.0 release of the compiler, it should port to other compilers with little difficulty.

PAUSE FOR MEMORY REFRESH

The windowing system described in these articles supports various video adapters, allowing 43- and 50-line screens on the EGA and VGA. It supports line wrap and scrolling in windows, boxed windows, etc., as well as basic editing features such as inserting and deleting characters and lines. The most important feature, however, is that the underlying text buffer (which I'll call the "window") can be much larger than the aperture into the text buffer that's displayed on the screen (the "viewport"). The text buffer can be made to slide horizontally or vertically beneath the viewport.

This feature is particularly useful in data-entry screens, which are, after all, what windows are all about. For example, you can have a 40-column-wide field in a data-entry screen that is actually a viewport into a 128-column window. When your typed text exceeds the 40 columns on the screen, it slides under the viewport as you add characters until the window's full width has been reached. The effect is similar to the horizontal-scrolling feature of most editors. You can also access multiple-line windows through a single-line viewport, using the cursor to bring hidden lines into view.

A SCREEN WITH A VIEW

This article presents the viewport component of the system—the code that draws the window on the screen. You can use these functions as a stand-alone—but primitive—window system.

The system starts with some macro definitions in VIEWPORT.H (Listing 1). The **#ifdef NEAR_HEAP** takes care of image management. As the viewport works, it needs to save two potentially large images: first, the underlying text must be saved when the viewport is created so that it can be restored when the viewport is deleted or moved. Next, the viewport needs to save a snapshot of itself when it is deactivated. This makes it possible to deactivate a window and then overwrite all or part of it with a second window. By deactivating this second window and reactivating the first, you can bring the first window back to the foreground without deleting the second. The reactivation subroutine uses the snapshot to redraw the window.

Having several windows on a VGA screen can use up quite a bit of memory. If **NEAR_HEAP** is defined, these images are saved in the standard, small-model heap. If **NEAR_HEAP** isn't defined, memory for the saved images is allocated from outside the program's data segment. This latter option lets you write a program in the small model, even if it uses several hundred kilobytes for window images. A similar mechanism (described in the previous article) holds the text buffers, which can also be quite large.

EDIFICE COMPLEX

The **viewport** structure is declared in Listing 1. The **magic** field is used to identify a valid, initialized **viewport** structure. It's set to **V_MAGIC** (defined just above the structure definition) when you initialize the **viewport**, and to **~V_MAGIC** when the **viewport** is discarded. All viewport functions and macros check this field before they manipulate the structure.

Allen Holub is a teacher, writer, consultant, and composer. He teaches C and compiler design for the U.C. Berkeley Extension. His most recent book is Compiler Design in C. *He can be reached at holub@violet.berkeley.edu (...!ucbvax!violet!holub) via USENET or c/o Software Engineering Consultants, P.O. Box 5679, Berkeley, CA 94705.*

ILLUSTRATION BY JIM CARPENTER

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The next two bit fields indicate whether the **viewport** is closed or inactive. A **closed** viewport has been removed from the screen, and its contents must be redrawn by the application program if it is reopened. A viewport must be closed in order to be resized. An **inactive** viewport has merely been deactivated, not closed. A subsequent reopen will restore it to its predeactivated condition. A deactivated **viewport** can be hidden (removed from the screen) while it is deactivated. Whether the viewport was hidden or visible at deactivation time, the **old_image** pointer at the bottom of the structure will be non-**NULL** and will hold a snapshot of the viewport.

The next two bit fields control how viewports are initialized. If **clearok** is set, the window is cleared when it is first opened (but not when it is reactivated). If **saveok** is set, the underlying text is saved (in a buffer pointed to by the **savebuf** field) when the viewport is first opened, and restored when the viewport is either closed or hidden.

Setting **saveok** to false will save some memory by not saving the text, but such a viewport cannot be moved without messing up the screen; an image of the viewport will remain on screen even after it is closed. These sorts of windows are useful, nonetheless, for subwindows, such as a field within a data-entry screen. The parent window is created first (and the underlying text is saved by it); then the subwindows are created on top of the parent without saving. The other fields in the **viewport** structure are self-explanatory.

MACRO MANIA

Listing 1 then continues with a host of macros that are used to manipulate the viewport. All are passed a **viewport** pointer as the first argument. The macro

```
v_valid(viewport v)
```

evaluates true if the argument points at a valid viewport (if its **magic** field holds a valid number).

A viewport is initialized to various defaults, which can be changed before it is displayed. The two macros

```
v_foreground(viewport v, int color)
v_background(viewport v, int color)
```

and the **v_saveok** and **v_clearok** macros discussed below typically are used after the **viewport** is initialized—but before it is displayed for the first time—in order to change the default settings. The current macros change the foreground and the background colors, respectively. The default colors are bright white characters on a black background. Other colors are defined in Borland's CONIO.H file, with symbolic names for the standard IBM character attributes. The new colors take effect immediately—any characters written to the viewport will have the new attributes. Existing characters are not redrawn, however: they remain in the original color unless they are overwritten. These macros do not test for a valid **viewport**-pointer argument.

A newly created window is not displayed until it is explicitly opened. A default window does not save text under the window when it is opened,

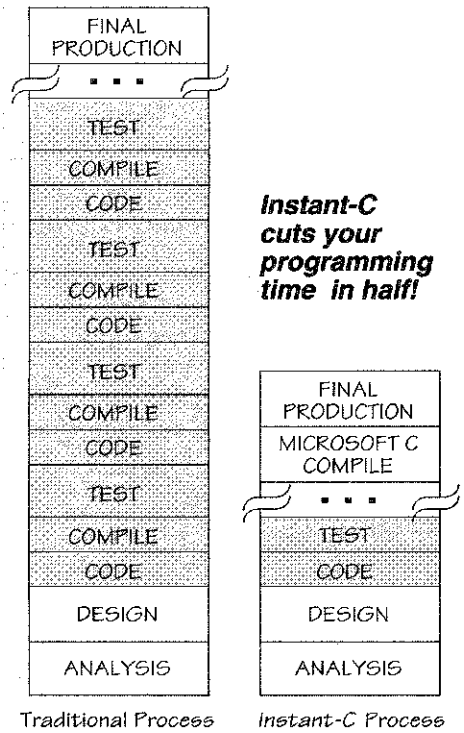
Turn your favorite C compiler into a powerful database manager with the
C/Database Toolchest
The C/Database Toolchest™ adds sophisticated file management functions to your Power C™, Turbo C®, QuickC®, or Microsoft® C compiler. With the C/Database Toolchest™, your data requires much less disk space than with programs like dBASE®, and you can access your data much faster. Of course the full power of C provides you with an unlimited amount of programming flexibility.
The C/Database Toolchest™ includes three major components:
1) An advanced B+tree library gives you instant access to your data.
2) A high-level ISAM library provides you with an easy-to-use C interface, and
3) A complete database manager (with C source code included) shows you how to create impressive applications.
You also receive a comprehensive 350 page manual and a utility for converting dBASE® files.
The C/Database Toolchest™ supports features that you'd expect to find only in products costing ten times as much. Advanced features include variable length records, variable length keys, multiple keys per index, and multiple indexes stored in a single file. Your data files can contain an unlimited number of records, and each record can be as large as 32K bytes in length.
About the only thing that the C/Database Toolchest™ doesn't do is cost you a lot of money. We've kept our price low so you can manage your budget as easily as your data.
Now Only $19.95!
Order Coupon
Name
Street
City
State Zip
Telephone
Paying By Money Order Check
Visa MC AX Disc.
Card#
Exp. Date
Disk Size 5¼" 3½"
Qty. Product Price Subtotal
C/Database Toolchest $19.95
C/Database Library Source B+tree & ISAM library source code $10.00
Add Shipping ($5 USA, $20 Foreign)
Texas Residents Add 8% Sales Tax
Total Amount of Your Order
Order now by calling our toll free number or mail the coupon to:
Mix Software
1132 Commerce Drive
Richardson, TX 75081
1-800-333-0330
60 Day Money Back Guarantee
Not Copy Protected ■ Royalty Free
For technical support, please call:
1-214-783-6001
MIX software
C/Database Toolchest and Power C are trademarks of Mix Software. QuickC and Microsoft C are registered trademarks of Microsoft. Turbo C is a registered trademark of Borland. dBASE is a registered trademark of Ashton-Tate.
Circle 43 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Double your C Productivity!

"A remarkable and utterly addictive speed improvement. "
– The C Gazette

The Instant-C™ development companion to Microsoft® C will cut your programming time in half. After coding, testing and debugging with Instant-C, optimize the final version of your code with Microsoft C. Instant-C saves you time by significantly shortening each coding and testing cycle and by reducing the number of cycles you need.



Traditional Process — Instant-C Process

Instant-C cuts your programming time in half!

Instant-C combines ultra-fast incremental compiling, flexible debugging, and compatibility with your current development tools making the perfect development partner for Microsoft C. Instant-C is compatible with libraries from Microsoft, Copia, Faircom, Metagraphics, Novell, Oakland, Raima, Roundhill, Vermont Creative, WNDX and other popular suppliers.

"Instant-C is fabulous for finding and fixing the sticky bugs."
– Richard Kasperski, The WNDX Corporation

Instant-C debugging finds bugs that other tools miss. It automatically detects over 500 errors, including runtime problems such as array references out-of-bounds, stack overflow, use of uninitialized pointers, overwriting a return address, read or write through null pointers and overwriting your program or DOS.

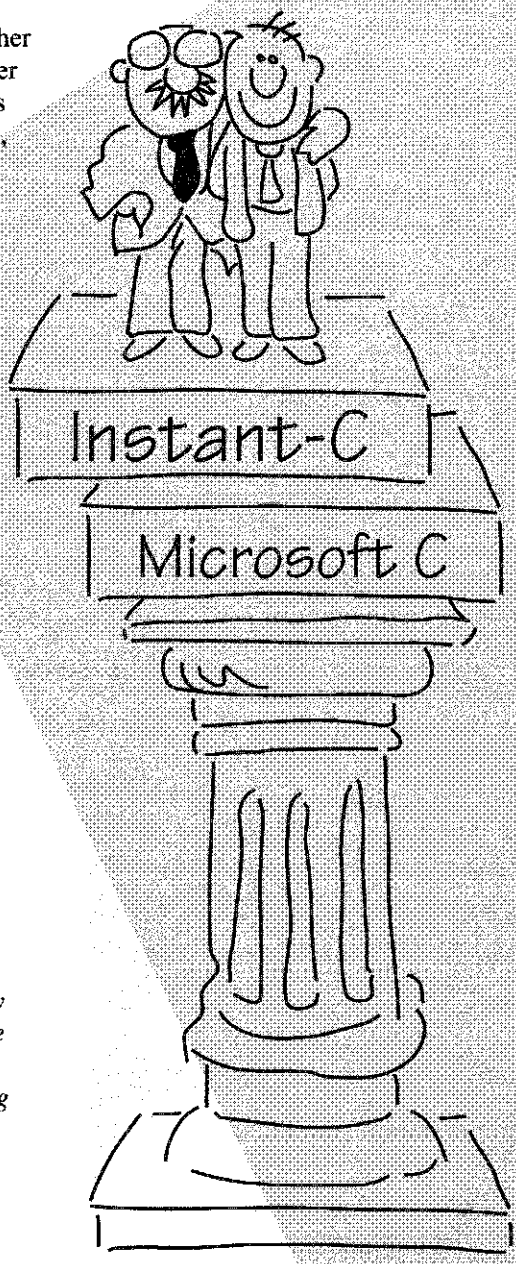
"Superb product! I simply won't code any other way."
– Michael Brooks, Enercalc, Inc.

Instant-C is ultra-fast. For example, when you make a change to your program, Instant-C recompiles and relinks just the change. In fact, Instant-C can rebuild a 100,000 line program after a small change in less than two seconds.

"How did we live without it?"
– Duane Burris, Mardian, Inc.

The Instant-C Environment:

- *The industry's fastest edit-compile-link-debug cycle*
- *Configurable built-in editor and source formatter*
- *Direct access to 16Mb of memory*
- *Thorough static and inter-module error checking*
- *Extensive run-time error checking*
- *Multiple virtual or real screens*
- *Powerful source-level debugging*
- *Support of commercial libraries*
- *Conditional breakpoints in C*
- *Compatibility with assembly language object modules*
- *Immediate evaluation of all C expressions, including macros*
- *Compatibility with Microsoft C compiler 5.1 and 6.0*
- *Interactive execution of functions*
- *Works with your favorite editor*
- *Allows use of Microsoft run-time libraries*
- *Dynamic cross-referencing and browsing*
- *Full graphics screen support*



Let Instant-C optimize your development... Try it today for only $495.00.
60 day money-back satisfaction guarantee.

1-800-666-2585

Instant-C™

Productivity Companion for Microsoft C

Rational Systems, Inc.

TEL: (508) 653-6006
FAX: (508) 655-2753

Circle 44 on Reader Service Card

© 1991 Rational Systems, Inc.
Instant-C is a trademark of Rational Systems, Inc. Microsoft is a registered trademark of Microsoft Corporation. All others are trademarks of their respective manufacturers.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

LISTING 1

```
/* viewport.h- viewport Definition and Macros */
#ifndef __VIEWPORT_H
#define __VIEWPORT_H

#include <alloc.h>              /* needed for farmalloc prototype */

#ifdef NEAR_HEAP
#       define VMALLOC  malloc /* Allocate memory for buffer.          */
#       define VFREE    free   /* Free memory used for buffer.         */
        typedef char    *cptr; /* pointer to a character buffer        */
        typedef int     *iptr; /* pointer to an int buffer             */
#else
#       define VMALLOC   farmalloc
#       define VFREE     farfree
        typedef char far* cptr;
        typedef int  far* iptr;
#endif

#define VMAXCOLS        80      /* Max. columns in a viewport           */
#define VMAXLINES       50      /* Max. lines (rows) in a viewport      */
#define VMAGIC          0xcdef  /* Magic number used to recognize a     */
                                /* valid viewport.                      */
#define VINIT_CHAR      ' '     /* clearok and v_scroll_region init.    */
                                /* viewport w/ this character           */
typedef struct viewport
{
    unsigned magic;         /* Magic number--identifies a valid viewport */

    unsigned inactive: 1 ; /* viewport is deactivated but still open    */
    unsigned closed  : 1 ; /* viewport is closed and deactivated.       */
    unsigned clearok : 1 ; /* Clear (blank fill) viewport on 1st open.  */
    unsigned saveok  : 1 ; /* Save area under viewport before displaying*/

    unsigned char nrows  ; /* number of rows in viewport                */
    unsigned char ncols  ; /* number of columns in viewport             */

    unsigned char row;      /* absolute coordinate of upper-left corner  */
    unsigned char col;
    unsigned char cur_row; /* cursor position     */
    unsigned char cur_col;

    unsigned char fcolor;  /* foreground color    */
    unsigned char bcolor;  /* background color    */
    iptr savebuf;          /* underlying text when window first opened  */
    iptr old_image;        /* old image when window closed or deactivated*/
}
viewport;
 /*-------------------------------------------------------------------------*/
#define v_valid(      v       ) ((v)->magic == VMAGIC)
#define v_foreground( v, color) ((v)->fcolor =  (color) )
#define v_background( v, color) ((v)->bcolor =  (color) )
#define v_saveok(     v, on   ) ((v)->saveok =  (on)!=0 )
#define v_clearok(    v, on   ) ((v)->clearok=  (on)!=0 )
#define v_getposn( rp, cp     ) ((*(rp)=(v)->row),  (*(cp)=(v)->col) )
#define v_row(      v ) (v_valid(v)? ( (v)->cur_row              ):-1)
#define v_col(      v ) (v_valid(v)? ( (v)->cur_col              ):-1)
#define v_nrows(    v ) (v_valid(v)? ( (v)->nrows                ): 0)
#define v_ncols(    v ) (v_valid(v)? ( (v)- >ncol         ): 0)
#define v_ateol(    v ) (v_valid(v)? ( (v)->cur_col== (v)- >ncols-1 ): 0)
#define v_atbol(    v ) (v_valid(v)? ( (v)->cur_col== 0          ): 0)
#define v_ateob(    v ) (v_valid(v)? ( (v)->cur_row== (v)- >nrows-1 ): 0)
#define v_advance( v ) v_gotorc( (v), (v)->cur_row, (v)- >cur_col + 1  )
#define v_backup(  v ) v_gotorc( (v), (v)->cur_row, (v)- >cur_col - 1  )
#define v_cr(      v ) v_gotorc( (v), (v)->cur_row     , 0            )
#define v_up(      v ) v_gotorc( (v), (v)->cur_row - 1, (v)->cur_col  )
#define v_down(    v ) v_gotorc( (v), (v)->cur_row + 1, (v)->cur_col  )

#define v_scroll(v,u,l) v_scroll_region(v,u,l,0,0,VMAXCOLS,VMAXLINES)
#define v_move_rel(v, dr, dc) v_move((v), (v)->row-(dr), (v)->col-(dc))
/*-------------------------------------------------------------------------*/
int  v_startup    ( int mode    );
void v_shutdown   ( void        );
int  v_construct  ( viewport *v );
int  v_destroy    ( viewport *v );
int  v_open       ( viewport *v );
int  v_close      ( viewport *v );
int  v_deactivate ( viewport *v, int hide );
int  v_size       ( viewport *v, unsigned nrows, unsigned ncols );
int  v_move       ( viewport *v, unsigned row,   unsigned col   );
int  v_gotorc     ( viewport *v, unsigned row,   unsigned col   );
int  v_putc       ( viewport *v, int c, int move_cur );
int  v_puts       ( viewport *v, char *s );
int  v_getc       ( viewport *v );
int  v_gets       ( viewport *v, int n,  char *s  );
int  v_scroll_region( viewport *v, int up, int to_left, int left, int top,
                                   int right, int bottom );
int  v_clear      ( viewport *v );
int  v_cleartoeol ( viewport *v );

/* Workhorse functions, not intended to be user callable*/

void _v_save_scr    (iptr image, int row, int col, int nrows, int ncols);
void _v_restore_scr (iptr image, int row, int col, int nrows, int ncols);
void _v_fill_scr    (int c_att,  int row, int col, int nrows, int ncols);
#endif /* #ifdef __VIEWPORT_H */
```

LISTING 2

```
/* Functions to start up and shut down the Video System,
   construct and destroy viewports */
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <tools/viewport.h>

static int Old_mode = LASTMODE; /* startup video mode, Return to this
                                 * mode when the window system exits.
                                 */
static unsigned Times_called = 0;

void v_shutdown ( void )
{
    /* V_shutdown shuts down the video system sometime before the
     * normal shutdown at program termination.
     */
    if( Old_mode != LASTMODE ){
        textmode( Old_mode );   /* Restore original mode */
        Old_mode = LASTMODE;
    }
    Times_called = 0;           /* allow a restart */
}

int v_startup( int mode )
{
    /* v_startup is passed either a video mode or LASTMODE, as defined in
     * <conio.h>, if mode==LASTMODE, the current video mode is used.
     *
     * v_startup calls atexit(), so the video system will shut down
     * automatically when the program exits. The video system can
     * be initialized only once. Subsequent calls do nothing & return
     * false unless the video system is shut down explicitly with a
     * v_shutdown() call.
     */
    struct text_info info;

    if( Times_called++ == 0){
        gettextinfo( &info );
        Old_mode = info.currmode;
        if( mode != LASTMODE )
            textmode( mode );
        atexit( v_shutdown );   /* Call v_shutdown on exit */
    }
    return( Times_called == 1 );
}
int v_construct( viewport *v )
{
    /* Initialize a previously allocated (with malloc() or a declaration)
     * viewport, setting various fields to reasonable defaults. It's an
     * error to construct a viewport more than once without an intervening
     * v_destroy call.
     */
    struct text_info info;
    gettextinfo( &info );

    if( v->magic == VMAGIC )
        return 0;

    v->magic     = VMAGIC;
    v->inactive  = 1;
    v->closed    = 1;
    v->clearok   = 0;
    v->saveok    = 0;
    v->nrows     = info.screenheight;   /* Initial size is entire screen */
    v->ncols     = info.screenwidth;
    v->row       = 0;                   /* Upper-left corner */
    v->col       = 0;
    v->cur_col   = 0;
    v->cur_row   = 0;
    v->fcolor    = BLACK ;
    v->bcolor    = LIGHTGRAY ;
    v->savebuf   = NULL;
    v->old_image = NULL;
    return 1;
}

int v_destroy ( viewport *v )
{
    /* Destroy the viewport, freeing all internally allocated memory, but
     * do not free the viewport itself.
```

continued

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

C CODE FOR THE PC

source code, of course

	MS-DOS File Compatibility Package (create, read, & write MS-DOS file systems on non-MS-DOS computers)	$750
	M++ (fast C++ math classes; indexing, matrices & vectors, numerics, memory handling, I/O; specify C++ compiler/Turbo, Zortech, Glock.)	$395
	C++/Views (C++ interface to MS Windows 3.0; over 60 classes; free broswer (no source); no royalties; Zortech C++ only)	$395
Upgrade!	GraphiC 6.0 (high-resolution, scientific plots in color & hardcopy, contour plots, device independence)	$370
Cheaper!	Victor Image Processing Library & ZIP Image Processor (brightness, contrast, merge images, TIFF/GIF/PCX/bin, ScanJet suppport)	$290
	TurboTEX (Release 3.0; HP, PS, dot drivers; CM fonts; LaTEX; MetaFont)	$250
	db_File & db_Retrieve Bundle by Raima (B-tree and network database with SQL query and report writer; multi-user $475)	$245
	TE Editor Developer's Kit for Windows (full screen editor, undo command, multiple windows; inc. TER for application build-in; no royalties)	$220
	Code Base 4 (database manager, dBase and Clipper compatibile indexes & data files; Version 4.2)	$210
	String Externalization Tools (one .exe, many languages; change language of prompts & messages by changing external file; no royalties)	$210
	Install 3.05 (automatic installation program; user-selected partial installation; CRC checking)	$195
NEW!	c_pslib (PostScript generation library for C programs; includes complete graphics, font, rotation & paragraph support	$170
	Report Right (report writer and graph generator for Windows 3.0)	$170
	C-Index/PC Database Library (fast B+ tree indexing, string/binary/custom keys, multiple record formats/file, 170-page manual)	$165
	Rogue Wave tools.h++ or math.h++ Class Library (extensive docs)	each $165
	DXF Subroutine Libraries (toolkit for making AutoCAD DXF drawing files; includes manual & sample programs)	$155
	WKS Library Version 2.01 (C program interface to Lotus 1-2-3, dBase, Supercalc 4, Quatro, & Clipper)	$155
	RED (program development editor for Microsoft Windows; keystroke recording & playback; unlimited undo)	$150
	Minix Operating System (Version 1.5; Unix-like operating system, includes manual; specify 5.25" or 3.5" diskettes)	each $150
NEW!	Delorie GCC for MS-DOS (Version 1.39; includes C++, assembler & DOS extender; complete source code and makefiles)	$150
	Vmem/C (virtual memory manager; least-recently used pager; dynamic expansion of swap file)	$140
	CBTree (B+tree ISAM driver, multiple variable-length keys)	$135
	TE Editor Developer's Kit (full screen editor, undo command, multiple windows)	$130
	C Communications Toolkit by Magna Carta (multi-port & co-processor board support, FAX, interrupt driven, emulations, xfer protocols)	$125
	C++ Object Library (virtual windows, I/O, lists, file free space management, keyed ISAM file I/O)	$125
	SilverWare "C" EMM Library (60 interface functions to EMM Version 4.0)	$120
	WinMem (unlimited global memory handles for Windows 3.0, 4-byte overheadper block rather than 20, virtual memory for 286)	$110
	Network Toolbox/N (C functions for Novell Netware 2.1x, sample programs included)	$110
NEW!	SCM (portable Scheme in C, conforms to IEEE and 3.99 specs, garbage collection off C stack for easy mixing with C)	$100
	PC/IP (CMU/MIT TCP/IP for PCs; Ethernet, Appletalk & NETBIOS drivers, RVD, Clarkson drivers, Stan's NFS clone, Bdale mailer)	$100
NEW!	Heapman (application memory management for Windows 3.0; 64K bytes of heap space; includes memory browser/debugger)	$100
	PowerSTOR (Version 1.2; extended heap space on extended memory, expanded memory, and/or hard disk)	$95
	MIRACL Multiprecision Integer & Rational Arithmetic C Library (nearly 100 functions; C & C++; from Ireland)	$90
	Disassembler (8086/80286/80386/80C186 with 80287/80387/80C187 NPX)	$90
	Booter Toolkit (floppy-disk bootstrap routines, DOS file system, light-weight multitasking, windows, fast memory management)	$85
	VM4C (virtual memory manager for C; swaps blocks to disk; small, fast, simple)	$80
Upgrade!	VMEM (virtual memory manager by Blake McBride, Version 3.6, LRU pager, dynamic swap file, image save/restore)	$80
	MultiDOS Plus (DOS-based multitasking, intertask messaging, semaphores)	$80
	BGI Printer Driver Toolkit (BGI drivers for Epson 9- & 24-pin, HP LaserJet II, & PaintJet printers)	$75
	COMM-DRV (interrupt-driven serial communication, device drivers & serial port monitors; source for libraries only; complete source $200)	$75
	FlexList (doubly-linked lists of arbitrary data with multiple access methods; specify C or C++)	each $65
	Foundations-1 C++ Class Library (ASCII record I/O, bit arrays, exception handling, B-tree, persistent objects)	$60
	MultiStr (evaluates a string containing multiple ASCII arithmetic expressions with variables)	$60
	Coder's Prolog (Version 3.0; inference engine for use with C programs)	$60
	FinanC (large collection of financial function including bond, inventory, stock portfolio, & cash flow)	$55
	Pascal P-Code Compiler & Interpreter or Pascal-to-C Translator (Level 0 ISO standard Pascal with some Level 1 features)	each $50
	CALC (ASCII algebraic expression evaluator, unlimited parenthesis nesting, symbols, 32 built-in functions, easily extended)	$50
	Backup & Restore Utility by Blake McBride (multiple volumes, file compression & encryption)	$50
	Floppy TAR (TAR backup and restore on MS-DOS devices; direct access to non-standard devices)	$50
	C Compiler Pack (5 C compilers; 3 for 8086 (gcc, MicroC, Small C), 2 for 68000 (Sozobon, cc68); gcc ports include library source only)	$50
NEW!	Mini IDL (interface generator for complex data; single inheritance, translator & table generator tools, ASCII external form))	$50
	GNUish MS-DOS (ports of 17 GNU programs to MS-DOS organized by Thorsten Ohl)	$50
	SuperGrep (exceptionally fast, revolutionary text searching algorithm; also searches sub-directories)	$50
	OBJASM (convert .obj files to .asm files; output is MASM compatible)	$50
	YATE (Yet Another Text Editor; written in C++, full-screen)	$50
	NIH Class Library & Book (basic C++ classes & *Data Abstraction and Object-Oriented Programming in C++* in softback by Keith Gorlen)	$50
	Kier DateLib (all kinds of date manipulation; translation, validation, formatting, & arithmetic)	$45
	RXC & EGREP Version 2.0 (Regular Expression Compiler and Pattern Matching; finite state machine from regular expression)	$35
	Bison & BYACC (YACC workalike parser generators; documentation; no restrictions on use of BYACC output)	$35
	PC-XINU (Comer's XINU operating system for PC)	$35
	Spell Pack (6 spelling programs, a hyphenator, 2 utility packs and a 60K word list: Ispell, Microsp, Sp, Cspella, Spell, Dawg, Soundex)	$30
	REGX Plus (search and replace string manipulation routines based on regular expressions)	$30
	GNU Awk & Diff for PC (both programs in one package)	$30
Upgrade!	6-Pack of Editors (ten public domain editors for use, study & hacking; includes microEmacs 3.10, Stevie, Elvis & Origami)	$30
	Crunch Pack (25 file compression & expansion programs)	$30
	Make Pack (eight versions of Make including GNU Make, Cake, and Gymake and a makefile maker)	$30
	BigNum (portable and efficient arbitrary-precision arithmetic package; hardcopy docs; from France)	$25
	String Pack (lots of string routines; C++ String class, BAWK, word wrap, fuzzy search, Boyer-Moore, Hypertext, etc.)	$25
	PC-MAIL (UUCP mailer by Wietse Z. Venema; send, receive, and manage UUCP mail; from Holland)	$25
	PERL for MS-DOS (C, sed, awk, and shell all rolled into one language; includes hardcopy docs)	$25
	FLEX (fast lexical analyzer generator; new, improved LEX; BSD Version 2.3.6 with docs)	$25
	Using C++ Library (the code from the book by Bruce Eckel and then some; Zortech 2.0 compatible)	$25
	GNU Chess (GNU Version 3.1 & Windows Version 1.01; executable included; requires SDK to build; truly beautiful!)	$20
	GNU RCS (FSF's version of the Revision Control System; like Unix's SCCS only better; keeps track of software development)	$20
	SDBM (fast, disk-based hash table manager for really large hash tables; clone of Unix *ndbm*)	$20
	VAL Linker by David Troendle (a complete linker; excellent for learning about linking)	$20
	NLMDL (non-linear least squares fitting in C++)	$20
	Unix/386	
NEW!	X-Windows (client & server, shared library, Xlib, executables, XAW etc.; diffs on X11R4; specify TCP or same machine)	$200
NEW!	Kyoto Common Lisp (Austin flavor, naturally; Version 1.530; includes GCC and Portable Common Loops (PCL))	$140
	GNU Emacs Version 18.55 (includes complete source & executables)	$125
	Waffle BBS (interface to UUCP & B or C news; internal forums; files section; external processing; full USENET support)	$120
	GNU C Compiler (gcc) Version 1.39 (includes complete source & executables)	$100
	ViewComp Spreadsheet (internal termcap; printer output; hardcopy docs)	$80
	GNU Debugger (gdb) Version 3.5 (includes complete source & executables)	$60
	Gillespie Pascal-to-C (many flavors of Pascal handled; readable & maintainable output)	$25
	Gosling Spreadsheet Pack (variants of the 1982 Gosling spreadsheet including PubliCalc and SC version 6.10; 1 MS-DOS variant)	$25

The Austin Code Works
11100 Leafwood Lane
Austin, Texas 78750-3587 USA

Voice: (512) 258-0785
FAX: (512) 258-1342
E-mail: info@acw.com

Free surface shipping for cash in advance **For delivery in Texas add 7%** **MasterCard/VISA**

Circle 45 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
    */

    if( v->magic != VMAGIC ){ return 0;  }
    if( !v->closed         ){ v_close( v );  }
    v->magic = ~VMAGIC;      /* guaranteed not to be the same as VMAGIC */
    return 1;
}
    /* Workhorse functions: Save, Restore, and Fill Screen Regions */
void _v_save_scr( iptr image, int row, int col, int nrows, int ncols )
{
    /* Copy the indicated region of the screen into image. */
    int i;
    int buf[ VMAXCOLS ], *bp;
    while( --nrows >= 0 ){
        gettext( col+1, row+1, col+ncols, row+1, buf );
        ++row;
        bp = buf;
        for( i = ncols ; --i >= 0 ; )
            *image++ = *bp++;
    }
}
void _v_restore_scr( iptr image, int row, int col, int nrows, int ncols )
{
    /* Like save_scr, but goes in the other direction */

    int i;
    int buf[ VMAXCOLS ], *bp;
    while( --nrows >= 0 ){
        bp = buf;
        for( i = ncols ; --i >= 0 ; )
            *bp++ = *image++ ;
        puttext( col+1, row+1, col+ncols, row+1, buf );
        ++row;
    }
}
void _v_fill_scr( int c_att, int row, int col, int nrows, int ncols )
{
    /* Fill the indicated region of the screen with c_att, which is
     * a character/attribute pair. row and col start at (0,0)
     */
    int i;
    int buf[ VMAXCOLS ], *bp;
    bp = buf;
        for( i = ncols ; --i >= 0 ; )
            *bp++ = c_att;
    for(; --nrows >= 0 ; ++row )
        puttext( col+1, row+1, col+ncols, row+1, buf );
}
int v_clear( viewport *v )
{
    _v_fill_scr( VINIT_CHAR | (v->bcolor << 4 | v- >fcolor) << 8,
                                        v->row, v->col, v->nrows, v->ncols );
    return 1;
}
int v_cleartoeol( viewport *v )
{
    _v_fill_scr( VINIT_CHAR | (v->bcolor << 4 | v- >fcolor) << 8,
        v->row + v_cur_row, v->col + v_cur_col, 1, v->ncols - v->cur_col );
    return 1;
}
```

LISTING 3

```
#include <stdio.h>
```

continued

Announcing version 2:

Victor Image Processing Library

Use Victor to develop powerful image applications

Work with images of any size -- use conventional, expanded, and extended memory

Now your applications can support 8-bit color and gray scale images of any size because Victor gives you complete control over conventional, expanded, and extended memory.

Image-based applications can be developed in MSC, QuickC, and Turbo C environments.

Display on Super VGA

Display images on EGA/VGA and super VGA up to 1024 x 768 256 colors.

Load & save PCX/TIFF/GIF/BIN

Handle images from any source, or create translation programs between the popular file formats.

Your application will be able to print halftones at any size.

Gray scale and color images

Powerful image processing for all images -- your software can have features like: zoom, resize, brighten, contrast, sharpen, outline, linearize, matrix conv, colorize, & more.

ScanJet and LaserJet support

You can offer device control for gray scale scanning -- AND print halftones at any size.

Victor supports Microsoft C, QuickC, and TurboC, includes demonstration and prototyping software, and full documentation. Source code available.

Give your applications support for scanner and laser printer.

Victor Library version 2, $195

Call (314) 962-7833 to order VISA/MC/COD

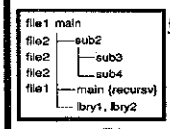
Catenary Systems 470 Belleview St Louis MO 63119 (314) 962-7833

Circle 46 on Reader Service Card

C AND C++ DOCUMENTATION

NEW! C-METRIC™ ($49) COMPLEXITY / QUALITY

- Calculates "Cyclomatic" path complexity for functions and system
- Counts Lines, Comments, Code, and 'C' Statements



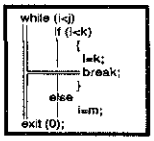
C-CALL™ ($59) FUNCTION HIERARCHY

- Tree-Diagram showing function hierarchy
- Table-Of-Contents of functions versus files
- Detailed cross-reference of functions

C-CMT™ ($59) FUNCTION COMMENT

- Generate/Insert function comment blocks
- Re-run anytime to update comment blocks
- Retains any user-generated comments

```
/*F*********************
sub2 USERS: main
     CALLS: sub3 sub4
     PARAM: arg1 arg2
     LOCAL: var1
     GLOBL: var2 var3
*********************F*/
```



C-LIST™ ($49) LIST OR REFORMATS

- Action-Diagrams show logic/control flow
- Reformats to various standardized formats
- Line numbers, page numbers, and titles

C-REF™ ($49) CROSS-REFERENCES IDENTIFIERS

- Summary list of Local/Parameter/Global/Define Identifiers used
- Detailed Cross-Reference of Identifiers vs Function/Line usage
- Produces Class-Hierarchy Tree-Diagram for C++ Classes

SPECIAL OFFER: ($189) Complete C-DOC™ Package

- All 5 programs fully integrated as 1 overall C-DOC program (DOS)
- NEW! special offer includes OS/2 protected-mode program
- Unconditional 30-DAY Money-Back Guarantee of Satisfaction !!

ACCURATE, and Fully AUTOMATIC (no source changes needed)

SOFTWARE BLACKSMITHS INC
6064 St Ives Way, Mississauga
ONT Canada L5N-4M1

VISA MasterCard
(416) 858-4466
Call for Immediate Shipment

Circle 47 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

nor does it clear the window (overwrite it with blanks) when it is opened. The two macros

```
v_saveok ( viewport v, int on )
v_clearok( viewport v, int on )
```

are used to change this behavior. Pass in a **true** second argument to enable the feature, a **false** to disable it. These macros do not test for a valid **viewport**-pointer argument.

MACROS ON THE MOVE

Some macros are used for positioning.

```
v_getposn( int *rp, int *cp )
```

tells you where the viewport is located on the screen. It loads ***rp** and ***cp** with the current row and column coordinates of the window's upper-left corner. The screen's upper-left corner is at (0,0).

To return the current viewport-relative cursor position, use

```
v_row(viewport *v)
v_col(viewport *v)
```

The upper-left corner of the viewport is (0,0). Both macros evaluate to –1 if **v** doesn't point at a valid **viewport**.

The following three macros

```
v_ateol(viewport *v)
v_ateob(viewport *v)
v_atbol(viewport *v)
```

evaluate true if the cursor is at the end of the line (touching the right edge of the viewport), at the end of the buffer (on the last line of the viewport), or at the beginning of the line (touching the left edge of the viewport). They all evaluate false if **v** doesn't point at a valid **viewport**.

The next group of macros enable cursor or window movement.

```
v_advance(viewport *v)
v_backup (viewport *v)
v_up     (viewport *v)
v_down   (viewport *v)
v_cr     (viewport *v)
```

The first four move the cursor one position to the right, left, up, and down, respectively; **v_cr** moves it to the left edge of the current line. The macros don't allow the cursor to go past the edge of the viewport. All evaluate true if the move is successful, or false if the move would cause the cursor to move outside the viewport.

```
v_move_rel(viewport *v, int dr, int dc)
```

moves the entire viewport some distance relative to the current position. **dr** ("delta row") is the distance along the vertical axis; **dc** ("delta column") is the distance along the horizontal. A negative **dr** goes down, a negative **dc** goes right. This rather counterintuitive setup lets the arguments match the **v_scroll** function (see below), which scrolls the text region within the viewport. That is, if you call **v_scroll** with the same second and third arguments as a **v_move_rel** invocation, the text moves within the viewport by the same amount and direction that **v_move_rel** moves the viewport as a whole.

STARTUP AND SHUTDOWN

The rest of Listing 1 contains prototypes for other library functions. On the code disk and over *PJ*'s BBS services, the functions are in separate files. However, to save space here (and to avoid printing **#include <stdio.h>** 20 times), the viewport functions have been merged into Listings 2–4.

The first two functions in Listing 2 start up and shut down the entire viewport system. The first is

```
int v_startup( int mode )
```

v_startup is passed either a video mode or LASTMODE (all defined in CONIO.H). If **mode** is LASTMODE, the current video mode is used; otherwise, the system flips into the new video mode. Available modes are

```
BW40, C40, BW80, C80, MONO, C4350
```

The names are self-explanatory, except for C4350, which gives you 43-line text mode on an EGA and 50-line text mode on a VGA. Usually, the video system is initialized only once. The first call to **v_startup** returns true. Subsequent calls do nothing and return false.

The video system shuts down automatically (and the old video mode is

NO SERMONS

If you're looking for a C/C++ magazine with lots of editorializing on what IBM might be doing or what Microsoft should have done — KEEP LOOKING.

But if you want a magazine with clear content, cogent code and penetrating coverage of relevant topics — LOOK NO FURTHER.

C GAZETTE

You simply won't find a better C/C++ magazine.

Call **800-234-0386** or FAX **503-746-0071** for a no-obligation, free trial subscription.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
#include <stdlib.h>
#include <conio.h>
#include <tools/viewport.h>

int v_open( viewport *v )
{       /*  Open (save underlying text and clear underlying region) the
            viewport.
     *  If clearok is enabled, area under viewport is filled with blanks
     *  the first time the viewport is opened, the behavior of subsequent
     *  calls depends on whether the viewport was closed or deactivated.
     *  If closed and clearok is enabled, the area is filled with blanks.
     *  If deactivated, the area is filled with the previous contents
     *  (at deactivate time). Underlying is saved if saveok is enabled
     *  and this is (1) a new or closed viewport or (2) a deactivated,
     *  hidden viewport. The text isn't saved for deactivated, but visible
     *  viewports. All the viewport-manipulation functions will open
     *  the window implicitly if an explicit open hasn't been performed.
     */
    if( v->magic != VMAGIC )
        return 0;

    /* If the window is closed, then it doesn't exist on the screen. Save
     * the underlying text and initialize if required. If it's inactive
     * but not closed, then it still exists on the screen and a refresh
     * is required.
     */
    if( v->closed || v->inactive ){
        if( v->saveok && !v->savebuf ){
            if( !(v->savebuf = VMALLOC(v->nrows * v- >ncols * sizeof(int))))
                return 0;
            _v_save_scr( v->savebuf, v->row, v->col, v- >nrows, v->ncols );
        }

        /* Initialize the viewport, either to the original image (before
         * it was deactivated) or to blanks (if clearok is true). If neither
         * condition is true, don't initialize. (I'm assuming that the
         * application will then fill up the window.)
         */

        if( v->inactive && v->old_image ){
            _v_restore_scr( v->old_image, v->row, v- >col, v->nrows,
                                                           v->ncols );
            VFREE( v->old_image );
            v->old_image = NULL;
        }
        else if( v->clearok )
            v_clear( v );
    }

    v->closed   = 0;
    v->inactive = 0;
    v_gotorc( v, v->cur_row, v->cur_col );
    return 1;
}
int v_close( viewport *v )
{
    /* Close the viewport (set it back to a pristine condition). The
     * underlying text is restored if it was saved by v_open. Any
     * old image (as is saved by v_deactivate()) is discarded.
     */

    if( v->magic != VMAGIC )
        return 0;

    if( v->old_image )                  /* If window is inactive,       */
        VFREE( v->old_image );          /* free the old image           */

    if( v->savebuf ){                    /* and restore original text.   */
        _v_restore_scr( v->savebuf, v->row, v->col, v- >nrows, v->ncols );
        VFREE( v->savebuf );
        v->savebuf = NULL;
    }
    v->closed   = 1;
    v->inactive = 1;
    return 1;
}

int v_deactivate( viewport *v, int hide )
{
    /* V_close() effectively sets the viewport back to its original,
     * pristine condition, as if it had never been opened. v_deactivate()
     * is less thorough. The contents of the window are saved in a local
     * buffer, and if hide is true, the original underlying text is
     * restored. A subsequent vopen() call restores the window to the
     * condition it was in before the vdeactivate.
     */

    if( v->magic != VMAGIC  ||  v->inactive )
        return 0;

    v->inactive = 1;

    if( !v->old_image )
        if( !(v->old_image = VMALLOC(v->nrows*v- >ncols*sizeof(int))))
            return 0;
    _v_save_scr( v->old_image, v->row, v->col, v- >nrows, v->ncols );

    if( hide && v->savebuf ){    /* Restore original text. */
        _v_restore_scr( v->savebuf, v->row, v->col, v- >nrows, v->ncols );
        VFREE( v->savebuf );
        v->savebuf = NULL;       /* Set savebuf null to tell v_open to */
    }                            /* save underlying text on reopen.    */
    return 1;
}

int v_size( viewport *v, unsigned nrows, unsigned ncols )
{
    /* Change the viewport size. The viewport must be closed
     * (not just deactivated) to change its size. False is
     * returned and nothing happens if the viewport isn't
     * closed, true otherwise.
     */
    if( v->magic != VMAGIC  ||  !v->closed )
        return 0;

    if( v->savebuf ){            /* A differently sized save buffer is   */
                                 /* required for the new size. Will      */
        VFREE( v->savebuf );     /* be allocated by v_open.              */
        v->savebuf = NULL;
    }
    if( v->old_image ){
        VFREE( v->old_image );
        v->old_image = NULL;
    }
    v->nrows = nrows;            /* Set up the new size. */
    v->ncols = ncols;

    return 1;
}

int v_move( viewport *v, unsigned row, unsigned col )
{
    /* Move the viewport to indicated absolute row and column. The
     * viewport must be closed or hidden to move it. If the viewport
     * is open and active, it's deactivated, moved, then reopened.
     * You may not move the viewport past the edge of the screen.
     * Return true if the viewport ended up where specified, false if
     * it bumped up against the side of the screen.
     */

    int rval       = 1 ;   /* return value */
    int was_active = !v->inactive;
    struct text_info info;

    if( v->magic != VMAGIC )
        return 0;

    gettextinfo( &info );

    if( row+v->nrows > info.screenheight  ){    /* Truncate row and col */
                                                /* so that the viewport */
        row = info.screenheight - v->nrows ;    /* stays on the screen. */
        rval = 0;
    }
    if( col+v->ncols > info.screenwidth ){
        col = info.screenwidth - v->ncols ;
        rval = 0;
    }
    if( was_active )
        v_deactivate( v, 1 );

    v->row = row;
    v->col = col;
    if( was_active )
        v_open( v );
    return rval;
}
```

LISTING 4

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <tools/viewport.h>
int v_gotorc( viewport *v, unsigned row, unsigned col )
{
    /* Move the cursor to the indicated viewport-relative row
     * and column. The cursor is not allowed to go past the edge
     * of the viewport. Nonetheless, it will go as far as possible
     * in the required direction. Return true if the cursor ended
     * up where specified, false if it didn't because it bumped
     * up against the side of the viewport.
     */
```

continued

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

restored) when the program terminates. A second function,

```
void v_shutdown ( void )
```

shuts down the video system before the normal shutdown at program termination. If needed, the video system can be restarted by a subsequent **v_startup**, but Borland's mode-change subroutine clears the screen as part of the start-up process.

CONSTRUCT AND DESTROY

A **viewport** structure is allocated using either a declaration or a **malloc** call. The structure is initialized (though not displayed) by a call to

```
int v_construct( viewport *v )
```

When the structure is displayed, it occupies the entire screen; the cursor is in the upper-left corner. Characters are white on black by default. You can change the defaults after the construction and before the open (which does the display) by using functions and macros such as the **foreground** and **background** macros discussed above.

The complementary subroutine

```
int v_destroy ( viewport *v )
```

removes the viewport from the screen (if it hasn't already been removed) and then frees all the internally allocated memory used for the viewport. It does not free the memory pointed to by the **v** argument, however (in case **v** points to a local variable rather than to **malloc**ed memory). **v_destroy** must be called before a local **viewport** goes out of scope.

SNAPSHOT ARRAYS

The remainder of Listing 2 contains the workhorse functions that save and restore snapshots of rectangular regions and fill a region with a character value. Although Borland's **gettext** and **puttext** functions provide these snapshots, they can't be used in a simple way, because the target buffer might be in far memory (if **NEAR_HEAP** was not defined when the modules were compiled). The solution is to use **gettext** or **puttext** to read from or write to a small buffer that holds a single line from the viewport. A small loop then copies this line-buffer's contents to or from the target buffer. Note that the snapshot arrays are arrays of **int**—not of **char**—because a screen character is a 16-bit quantity; the ASCII value is in the low byte, and a color-specification attribute is in the high byte.

The final function in Listing 2

```
int v_clear( viewport *v )
```

clears the viewport (fills it with blanks).

VIEWPORT HIDE AND SEEK

Listing 3 contains the bulk of the other viewport functions. The

```
int v_open( viewport *v )
```

function displays the viewport on the screen. The **v_open** function is passed a pointer to an initialized **viewport**, which might have been modified by invocations of macros such as **saveok** or **foreground.** You can also call this function to reactivate a deactivated window. All the window I/O and cursor-positioning routines and macros call **v_open** implicitly, so an explicit **v_open** call is required only to display a window or bring it to the foreground without modifying it.

If **clearok** is enabled, the area under the viewport is filled with blanks the first time the viewport is opened. The behavior of subsequent **v_open** calls depends on whether the viewport was closed or was deactivated. If it was closed and if **clearok** is enabled (by invoking the **clearok** macro on the current viewport), the area is filled with blanks on reopening. If the viewport was deactivated, the area is filled with the previous contents (as stored at deactivate time).

The underlying text is saved before the region is overwritten, but only if **saveok** is enabled and if one of the following applies: this is a new or previously closed viewport, or this is both a deactivated and hidden viewport. The underlying text is not saved for deactivated (but visible) viewports;

NO TUTORIALS

If you're looking for a C/C++ magazine with a lot of beginner C tutorials or a bunch of long-winded academic back-patting — KEEP LOOKING.
But if you want a magazine that's code-intensive, in-depth and packed with C/C++ source code — LOOK NO FURTHER.

You simply won't find a better C/C++ magazine.

Call **800-234-0386** or FAX **503-746-0071** for a no-obligation, free trial subscription.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

rather, the save buffer from the original **v_open** remains intact.

The next function

```
int v_close( viewport *v )
```

closes the viewport and resets it as if it had never been opened. If the underlying text was saved by **v_open,** this function also restores that text. Note that the condition of **saveok** at open time controls whether or not the text is restored when the window is closed.

Deactivate a window by calling

```
int v_deactivate( viewport *v, int hide )
```

A deactivated window can also be closed, in which case the memory used for the viewport snapshot is freed. **v_deactivate** is less thorough than **v_close**. A snapshot of the current contents of the viewport is saved in a local buffer, and if the **hide** argument is true, the original underlying text is restored. The original text is not restored when **hide** is false. A subsequent **v_open** call restores the viewport to its condition before **v_deactivate** was called. If the viewport is hidden, the underlying text is saved again at that time, so text that is written to the screen after the window is hidden is saved correctly when the window is redisplayed.

You can move a deactivated window—in fact, the window must be deactivated before it can be moved. (The move function does this automatically.) The screen will look strange if you move a deactivated (but visible) viewport, however, because the old image will still be on the screen.

Note that, as with **v_close,** the status of **saveok** at open time controls whether or not the underlying text is restored when the viewport is hidden. The **saveok** status at deactivation time is immaterial.

MOVERS AND SHAPERS

The next few functions manipulate viewports. The first is

```
int v_size( viewport *v, unsigned nrows,
            unsigned ncols )
```

This function changes the size of a viewport. It is usually called at least once before the initial display—after the **v_construct** call but before the **v_open**—in order to size the window down from the entire screen. You can call the function at other times, but the window must be closed (not just deactivated). If the window is open when you call **v_size,** it returns an error condition (0) and does nothing.

You can change the position of the viewport on the screen with a call to

```
int v_move( viewport *v, unsigned row,
            unsigned col )
```

The viewport should be either closed

```
    if( v->magic != VMAGIC )
        return 0;

    if( v->inactive )
        v_open( v );

    v->cur_row = min( row, v->nrows - 1 ); /* truncate to fit in window */
    v->cur_col = min( col, v->ncols - 1 );

    window( v->col+1, v->row+1, v->.>col+v->ncols, v- >row+v->nrows );
    gotoxy( v->cur_col + 1, v->cur_row + 1 );

    return (v->cur_row == row  &&  v->cur_col == col);
}
int v_putc( viewport *v, int c, int move )
{
    /* Overwrite the character on which the viewport's cursor is
     * resting. The cursor position advances one notch if the move
     * argument is true. It won't move past the right edge of the
     * line, however. Note that all characters are printed literally,
     * \n yields a graphics character, it doesn't go to the next line.
     * Return true if the cursor moved (it won't at end of line),
     * false otherwise.
     */

    int outc;
    int col, row;

    if( v->magic != VMAGIC )
        return 0;

    if( v->inactive )
        v_open( v );

    /* output the character. Can't use putch() both because we
     * don't have enough control of the color and because it
     * screws up the cursor if we're writing to the bottom right
     * corner of the screen.
     */

    outc = (c & 0xff) | ((v->bcolor << 4 | v->fcolor) << 8);
    col  = v->col + v->cur_col + 1;
    row  = v->row + v->cur_row + 1;
    puttext( col, row, col, row, &outc );

    return( move ? v_advance(v) : 0 );
}
int v_puts( viewport *v, char *s )
{
    /* Write string to current cursor position. Only that part of the
     * string that will fit on the current line is written. The cursor
     * moves to the character following end of string if it can. If the
     * output string was truncated, it will rest at the far right of the
     * line.  Return false if the string won't fit on the line, so was
     * truncated. Return true otherwise.
     */

    if( v->magic != VMAGIC )
        return 0;
    while( *s )
        if( !v_putc(v, *s++, 1) )
            return 0;
    return 1;
}

int  v_getc( viewport *v )
{
    /* Return the character at the current cursor position in the
     * indicated window, which is activated if necessary. Cursor
     * position doesn't change.
     */

    int c;
    int col, row;

    if( v->magic != VMAGIC )
        return 0;

    if( v->inactive )
        v_open( v );

    col  = v->col + v->cur_col + 1;
    row  = v->row + v->cur_row + 1;
    gettext( col, row, col, row, &c );

    return c & 0xff ;   /* strip attribute when returning */
}

int v_gets( viewport *v, int n, char *s )
{
    /* Copy at most n-1 characters into s, starting at current cursor
     * position into s. Return number of characters read.
     */

    char *start = s;

    if( v->magic != VMAGIC )
        return 0;

    while( --n > 0 ){
        *s++ = v_getc( v );
        if( !v_advance(v) )     /* then we just read the last character */
        {                       /* on the line                          */
            *s = '\0';
            return 0;
        }
    }
    *s = '\0';
    return (s - start);
}
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

or hidden when moved. As with **v_size**, **v_move** is usually called after the viewport is constructed by **v_construct**, but before it is displayed by **v_open**. If it is open and active, the viewport is deactivated, moved, and then reopened. The routine won't let you move the viewport outside the screen. If you try, the viewport will be placed as close as possible to the specified, illegal position. **v_move** returns true if the viewport ended up where you specified, or false otherwise.

The final movement function is

```
int v_scroll( viewport *v, int up, int left )
```

(I'll include the code for this next time; for now it's on the code disk and BBSs only.) This function scrolls the text in the viewport horizontally or vertically (or both) without moving the viewport itself. The text is moved up by **up** lines and left by **left** lines. The opened-up regions are filled with spaces. The **up** and **left** arguments may be zero or negative. The text moves right if **left** is negative, and down if **up** is negative. The cursor position is not changed by the scroll. This function cannot clear the viewport—if the requested movement is greater than the viewport dimension, **v_scroll** returns false and no movement occurs. To clear, use **v_clear.**

I/O FUNCTIONS

The I/O functions (Listing 4) manipulate the cursor and execute input and output to the viewport. A call to

```
int v_gotorc( viewport *v, unsigned row,
              unsigned col )
```

will position the cursor at the desired viewport-relative row and column (the upper-left corner of the viewport is row zero, column zero). The cursor won't go past the edge of the viewport, however; as with **v_move**, it will go to the nearest legal position. The subroutine returns true if it put the cursor at the specified location, false otherwise. Note that passing in very large **row** and **column** arguments is a convenient way to get to the viewport's bottom-right corner. Just ignore the error-return code.

Two output functions are provided.

```
int v_putc( viewport *v, int c, int move )
```

overwrites the character at the cursor position. The cursor then advances one notch if the **move** argument is true (but it won't move past the right edge). Note that all characters are printed literally. For example, '**\n**' yields an IBM graphics character—it doesn't go to the next line. **v_putc** returns true if **move** is true and if the cursor moved. Otherwise, it returns false.

The function

```
int v_puts( viewport *v, char *s )
```

writes an entire string to the viewport, starting at the current cursor position, though not past the right edge. If the string is longer than the available space, the extra characters are not printed. False is returned if the output string was truncated, true if it was not.

The final two routines in Listing 4 provide input. The function

```
int  v_getc( viewport *v )
```

returns the character at the current cursor position. The cursor is not moved. The function

```
int v_gets( viewport *v, int n, char *s )
```

copies at most **n**–1 characters into the string pointed to by **s**, starting at the current cursor position and stopping when **n**–1 characters are copied or when the cursor is against the right edge. The routine returns the number of characters actually read.

SO LONG FOR NOW

The code disk for this issue contains a program to test the functions and a makefile for use with the make utility that comes with Borland's compiler. This makefile can be used to create the VIEWPORT.LIB library, which holds all the viewport functions.

That's the display side of things. As you can see, these functions are useful and are, by themselves, adequate for many simple window applications. Next time I'll integrate them with the text-manipulation routines to create a truly useful windowing system.

NO VAPORWARE

If you're looking for a C/C++ magazine that advertises products months (even years) before they're available — KEEP LOOKING. But if you want a magazine that only accepts ads for existing products, and has independent, hard-hitting reviews — LOOK NO FURTHER.

C GAZETTE

You simply won't find a better C/C++ magazine.

Call **800-234-0386** or FAX **503-746-0071** for a no-obligation, free trial subscription.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Loop unrolling: A RISCy business

A look at a powerful scalar optimization on RISC and CISC processors

By Mark J. Barrenechea

Amdahl's law states that a program's total execution time is rate-limited by its slowest portion—or rather, by a compiler's worst optimization. Because existing programs are usually scalar-bound, not vector-bound, scalar optimizations ("software pipelining" and "loop unrolling") are as important as vector optimizations ("vectorization"). That is to say, vectorized code is still rate-limited to the surrounding scalar code. Scalar and vector optimizations were once restricted to mainframes and supercomputers, but ever since the Intel i860 became popularized on the PC/AT architecture, these optimizations have been migrating to desktop compilers. Vectorization is discussed here solely in terms of its usefulness in defining and clarifying scalability. Instruction scheduling, even though its behavior is synergistic with unrolling, is an important enough topic to be given its own article at a later date. This article will examine only loop unrolling.

THE RISC ARCHITECTURE

Understanding the impact of software pipelining and loop unrolling is inseparable from understanding the RISC architecture—in particular, the i860 microprocessor. The i860 chip has multiple execution units: an integer core unit, a floating-point unit, and a graphics unit. The *integer unit* is the i860 workhorse. It contains the integer registers, fetches both integer and floating-point instructions, and decodes and executes load, store, integer, bit, and control-transfer operations. The *floating-point unit* contains both the floating-point adder and the floating-point multiplier. The adder performs floating-point addition, subtraction, comparison, and conversions. The multiplier performs floating-point and integer multiply and floating-point reciprocal operations. The *graphics unit* has a special 64-bit integer logic that allows it to operate on long integers, along with a set of special graphics instructions for 3-D algorithms.

The i860 can execute one instruction per cycle as long as there are no register (source vs. destination) conflicts between adjacent instructions. The i860 processor also has a dual-instruction mode that can simultaneously ("in parallel") execute an instruction on any of the three execution units. Additionally, a pair of instructions can execute simultaneously within the floating-point unit, because the adder and multiplier units can operate in parallel. Thus, the i860 can execute as many as four instructions in one clock cycle. The art of keeping these instructions executing harmonically—one per cycle, or optimally, as many as four per cycle in dual-instruction mode—is called *software pipelining*. Figure 1 shows a series of operations that can be performed either in scalar or vector ("pipeline") mode [Reference 1].

In the scalar example, one instruction per cycle is executed (i.e., five instructions in five cycles). No delays are encountered because there are no register conflicts between adjacent instructions. The vectorized sequence executes eight instructions in two cycles by using the adder pipeline. Note, however, that code size increases 62.5% (eight vs. five instructions).

WORKING DEFINITIONS

No agreed-on definitions exist for scalability and vectorization. In the absence of any rigorous definition, categorizing a process as "scalar" or "vectored" is about as descriptive as calling it "modern" [2]. A useful working definition—based on current compiler methodologies—is this: the way a compiler translates high-level source code into assembly instructions determines whether a program has been vectorized or scalar-optimized.

On architectures such as the Intel i860's, there are no explicit vector instructions or registers. On other architectures, such as Cray's, there exist multiple register banks (scalar and vector) and multiple instruction sets (scalar and vector). Yet both types of architectures share similar vector characteristics. They have in common a multistage pipeline that allows for simultaneous instruction execution. Additionally, the i860 can emulate

Mark J. Barrenechea is vice-president of language development at MicroWay, Inc., in Plymouth, Massachusetts.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

high-speed vector registers by keeping variables in its high-speed data cache. The i860 can thus be categorized as a "vector" processor.

Any situation (sequence of code) that a compiler can recognize that uses this pipeline efficiently is considered a vectorized optimization. Any situation a compiler can recognize that uses any other architectural feature—such as the removal of branches or the reordering of instructions—is considered a scalar optimization. Ideally, a compiler automatically accepts code that was written in a higher-level language such as Fortran or C and translates it into vector (pipelined) instructions for the target machine without any need for programmer intervention. This can be achieved using a preprocessor, such as Pacific Sierra's Vector and Array Syntax Translator (VAST), or a front-end compiler flow analysis, as in MicroWay's new vectorizing compilers. Examining the benefits and drawbacks of each approach would require a lengthy discussion.

It's important to identify the most time-consuming portions of an application and then concentrate on optimizing them. These "hot spots" are often found in loops. Consider a simple example:

```
      do 10 i = 1, 50
         a(i) = b(i) + c(i)
         d(i) = a(i) / 2.0
10    continue
```

and its equivalent Fortran 90 array assignment syntax:

```
a(1:50) = b(1:50) + c(1:50)
d(1:50) = a(1:50) / 2.0
```

This array syntax adds the elements 1 through 50 of **b** and **c** and stores the results in elements 1 through 50, respectively, of **a.** Then the elements 1 through 50 of **a** are divided by 2.0, and the results are stored in elements 1 through 50, respectively, of **d.**

Both scalar and vector code can be generated from this loop. In scalar mode, a compiler can unroll the loop, schedule the instructions, and gain scalar improvements (see Livermore results, below). In vector mode, the loop can be transformed into a series of function calls

```
vadd( a[1], b[1], c[1], 50 )
vdiv( d[1], a[1], 2.0, 50 )
```

to routines that contain highly optimized "vector/pipelined" code for doing vector adds (**vadd**) and divides (**vdiv**). The

FIGURE 1: Operations in scalar and vector mode

Pseudo Operations:

```
f18:f19  <---  f6 + f7
f18      <---  f2 + f3
f17      <---  f5 - f4
f22      <---  f12 - f13
f20:f21  <---  f18:f19 - f10:f11 (where f18:f19 = f6+f7)
```

Scalar Code:

```
fadd.sd     f6, f7, f18;        f18:f19  <---  f6 + f7
fadd.ss     f2, f3, f16;        f16      <---  f2 + f3
fsub.ss     f5, f4, f17;        f17      <---  f5 - f4
fsub.ss     f12, f13, f22;      f22      <---  f12 - f13
fsub.dd     f18, f10, f20;      f20:f21  <---  f18 - f10
```

Vector (Pipelined) Code:

		Adder Pipeline Result		
Instruction	**Stage 1**	**Stage 2**	**Stage 3**	**Result**
pfadd.sd f6, f7, f0	[f6+f7]	?	?	0
pfadd.ss f2, f3, f0	[f2+f3]	[f6+f7]	?	0
pfsub.ss f5, f4, f0	[f5-f4]	[f2+f3]	[f6+f7]	0
pfsub.ss f12, f13, f18	[f12-f13]	[f5-f4]	[f2+f3]	f18:f19=[f6+f7]
pfsub.dd f18, f10, f16	[f18-f10]	[f12-f13]	[f5- f4]	f16=[f2+f3]
pfadd.ss f0, f0, f17	0	[f18-f10]	[f12- f13]	f17=[f5-f4]
pfadd.ss f0, f0, f22	0	0	[f18-f10]	f22=[f12-f13]
pfadd.ss f0, f0, f20	0	0	0	f20:f21=[f18-f10]

ILLUSTRATION BY JIM CARPENTER

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

TABLE 1: i860 unrolling results

Unroll Factor	Harmonic Mean	Median Rate	Code Size	Compile Speed
0	1.67	1.88	143,000	4:35
2	4.97	5.90	150,000	5:40
4	5.02	6.32	154,000	7:01
8	**5.24***	6.89	161,000	9:10
16	5.24	**7.28***	175,000	15:77
32	5.09	6.86	207,000	26:88
64	4.86	6.53	268,000	56:12

Harmonic mean and median rate are in MFLOPS.
Code size is in bytes.
Compile speed is in minutes:seconds
*Bold values represent the best number for the given unroll factor.

AN ACCURATE MEASURE

An average of execution times that measures total execution time is called the arithmetic mean. Consider

$$1/n \sum_{i=1}^{n} \text{Time}_i$$

where Time_i is the execution time for the ith program in a series of n. If performance (the number of floating-point operations performed) is expressed as a rate (such as MFLOPS), then the average measure of the total execution time is called the harmonic mean. Consider

$$n/\sum_{i=1}^{n} 1/\text{Rate}_i$$

where Rate_i is a function of $1/\text{Time}_i$, the execution time for the ith program in a series of n.

The harmonic mean is an accurate measure for Livermore loops in scalar mode [7]: each kernel can be optimized in some manner, and no weighting mechanism is needed to scale the numbers between the optimized and nonoptimized loops. When Livermore loops are vectorized, some kernels cannot be vectorized with current vector technology. Thus, the harmonic mean significantly understates the kernel performance. A weighted harmonic mean is needed to judge vectorized Livermore loops accurately.

compiler could, of course, in-line this code, but because of implementation restrictions it is easier to generate function calls. **vadd** and **vdiv** are called *idioms*—that is, routines found in a vector library. Examples of idioms are dot products and vector arithmetic operations. A compiler recognizes a finite set of these idioms and generates the appropriate sequence of library calls.

In general, when code can be vectorized, vectorization produces faster-executing code than scalar code for any given "hot spot." This will be demonstrated below.

THE PARADIGM

In order to measure the benefits of loop unrolling, a paradigm is needed to compare performance. Livermore Loops are 24 sample routines that emulate typical numeric-intensive programming. Although these routines may not represent real-world programming, they do provide an industry-accepted yardstick by which to measure performance. A brief explanation of each routine can be found on the code disk for this issue.

To focus only on the performance changes attributable to unrolling, superfluous optimizations—such as in-lining, loop rotation, and local and global common-subexpression elimination—have been turned off for these tests.

Although it's difficult to judge the overall effectiveness of a benchmark using a single number, a measure is needed to evaluate the scalar effectiveness. Within Lawrence Livermore loops, one could use many different numbers (geometric mean, harmonic mean, median rate, etc). The harmonic mean (see "An Accurate Measure," at left), a reasonable compromise in scalar mode, is used here [3].

AROUND THE LOOP AND BACK

Loop unrolling is easy to understand. The loop is reorganized as follows:

- The loop body is replicated n times, where n is the number of times the loop is unrolled (typically, four or eight).
- Any index register increment/offset instructions are eliminated from all replicated blocks except the last one. The original increment is multiplied by n. Any load-and-store instructions are adjusted accordingly.
- The code that handles the loop-counter is eliminated from all replicated blocks except for the last one. The original increment is multiplied by n.
- Branch instructions are eliminated from all replicated blocks except the last [4].

Three performance benefits of loop unrolling become evident:

- A significant number of branches are eliminated.
- Redundant instructions between nested loops may be eliminated, reducing the total number of instructions executed.
- If the loop body is small, all or most loop overhead may be reduced.

Eliminating or significantly reducing loop overhead increases loop execution speed.

Other benefits are less obvious. Because a loop is unrolled n times, there are more instructions (n times the number of blocks) to schedule. This improves the likelihood of reducing scheduling delays. This side effect has a great impact on RISC architectures, but less on CISCs such as the Intel 80486. In a pipeline architecture, the penalty paid for control of transfer instructions can be expensive, because a transfer breaks the flow of execution in the execution pipeline. Loop unrolling has three small disadvantages: the increased compilation time, the growth in code size ([loop body size] * [unroll factor −1]), and the complexity of debugging an unrolled loop (see Table 1).

OTHER ISSUES TO PONDER

In determining the efficiency of loop unrolling, two other architectural features must be considered: the number of available registers and the instruction cache size. By itself, unrolling has an innocuous effect on register coloring. However, when coupled with instruction scheduling, unrolling a loop may separate load-and-store operations, prolonging the lifetimes of certain variables that would otherwise die. This separation may also cause the variable to be either colored to different registers (because most architectures have a limited number of registers) or be spilled, thus "deoptimizing" the unrolled loop. The total effect of unrolling and instruction scheduling on register coloring is not yet known.

The instruction cache size must also be considered. What happens if the unrolled loop becomes larger than the instruction cache? Because the loop must be reloaded

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

into the instruction cache for each iteration, perhaps the loop should not be unrolled at all. The performance degradation caused by cache misses may outweigh the advantages of unrolling the loop. This impact is minimal on the i860, because refreshing the instruction cache incurs only a one-cycle delay.

UNROLLED EXAMPLES

Loop unrolling (also called *loop linearization*) is demonstrated in the following example, in which:

```
      DO  10  I = 1, 4
         X(I) = I
10    CONTINUE
```

is unrolled four times:

```
      X(1) = 0
      X(2) = 1
      X(3) = 2
      X(4) = 3
```

All loop overhead in this manufactured example has been removed.

Let's consider a more representative program, DAXPY (the name stands for DA times X Plus Y). The key routine used by Gaussian elimination (in the Linpack benchmark) is DAXPY [5]. Consider

```
      DO  10  I = 1, N
              Y(I) = Y(I) + DA * X(I)
10       CONTINUE
```

and its unrolled equivalent:

```
      J = N - MOD(N,4)          ! 4 is the unroll
                                  factor
      DO  10  I = 1, J, 4
         Y(I+0) = Y(I+0) + DA * X(I+0)
         Y(I+1) = Y(I+1) + DA * X(I+1)
         Y(I+2) = Y(I+2) + DA * X(I+2)
         Y(I+3) = Y(I+3) + DA * X(I+3)
10       CONTINUE
```

For any **N** that is divisible by 4 (that is, the unroll factor), rolled and unrolled loops process the same number of array elements. For any **N** that is not divisible by 4, **MOD(N,4)** elements are not accessed. For example, if **N** = 10, two array elements are not accessed in the unrolled portion.

Finally, let's consider a complicated nested **do** loop that cannot be unrolled or vectorized because of the coding style [6]:

```
      DO  10  J = 1, 4
         DO 20  I = 1, N
            C(J,I) = 0.0
20          CONTINUE
         DO 10  K = 1, 4
            DO 10  I = 1, N
               C(J,I) = C(J,I) + A(J,K) * B(K,I)
10       CONTINUE
```

The loop can be optimized in both scalar and vector modes by restructuring the loop as follows:

TABLE 2: Unrolling statistics for the i860

	UNROLLED						
KERNEL	0	2	4	8	16	32	64
1	3.37	10.19	9.95	10.95	**11.23***	9.32	9.12
2	2.44	6.91	6.77	6.91	6.91	6.91	**7.22***
3	2.53	10.73	11.26	11.85	**12.61***	11.85	6.26
4	2.88	8.40	9.13	9.55	9.55	9.55	**10.00***
5	1.75	5.32	5.21	6.41	7.81	8.62	**8.93***
6	1.79	4.02	4.02	**4.13***	**4.13***	4.02	3.82
7	4.21	10.61	10.47	**10.76***	10.61	**10.76***	10.61
8	2.15	7.89	7.89	7.89	7.89	7.89	**7.96***
9	3.08	**8.49***	8.31	8.40	8.40	8.40	8.40
10	0.95	**4.55***	4.44	4.49	4.49	4.49	4.49
11	1.23	4.58	5.73	6.88	7.64	**8.09***	7.64
12	1.29	5.55	6.52	7.49	**8.33***	**8.33***	**8.33***
13	0.96	2.40	2.40	**2.43***	**2.43***	2.40	2.37
14	1.02	**2.62***	**2.60***	**2.62***	**2.62***	**2.62***	**2.62***
15	0.86	**2.99***	**2.99***	**2.99***	**2.99***	**2.99***	**2.99***
16	2.21	**4.14***	3.94	**4.14***	**4.14***	4.04	4.04
17	2.67	**9.25***	9.04	**9.25***	**9.25***	**9.25***	**9.25***
18	1.71	**7.07***	6.98	**7.07***	6.89	6.81	6.81
19	2.05	7.58	7.39	**8.21***	7.77	4.41	3.56
20	2.85	**6.25***	6.13	**6.25***	5.16	5.16	5.16
21	1.18	**3.08***	**3.08***	**3.08***	3.06	2.94	2.94
22	1.56	**3.15***	3.11	**3.15***	**3.15***	3.11	**3.15***
23	1.97	8.51	8.51	**8.64***	**8.64***	**8.64***	8.61
24	1.42	3.29	3.47	**3.91***	3.68	3.68	2.89

*Bold values are the best for a given kernel.
All units are in MFLOPS.

Break Through
The 640k Barrier

with Memory
Management Libraries
fromTHE TOOL MAKERS

PowerSTOR® **$119.95**
Configure Expanded Memory, Extended Memory or Hard Disk as virtual memory. Allocations from 1 byte to 1 gigabyte. Ideal for data array structures. Supports most standard C compilers.

The Heap Expander® **$79.95**
Access Expanded Memory or Extended Memory. Swaps to Hard Disk as needed. Ideal for link list data structures. Supports Microsoft C and Turbo C.

Call Toll Free to order

USA
800-248-1045
ext 100

For more information, call The Tool Makers direct.

THE TOOL MAKERS
P.O. Box 2151
Santa Cruz, CA 95063
408-458-0690

Circle 48 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

FIGURE 2: i860 optimizations for Lawrence Livermore loops

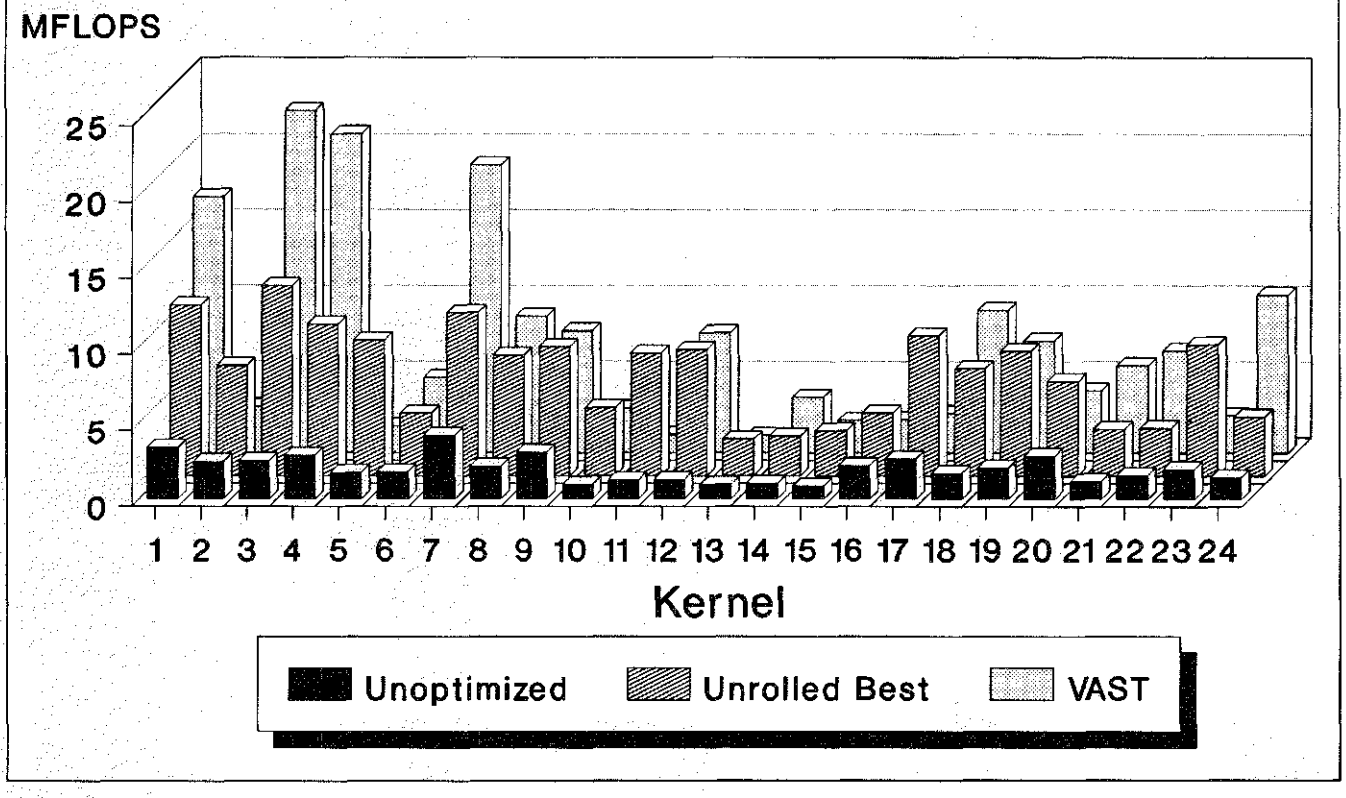
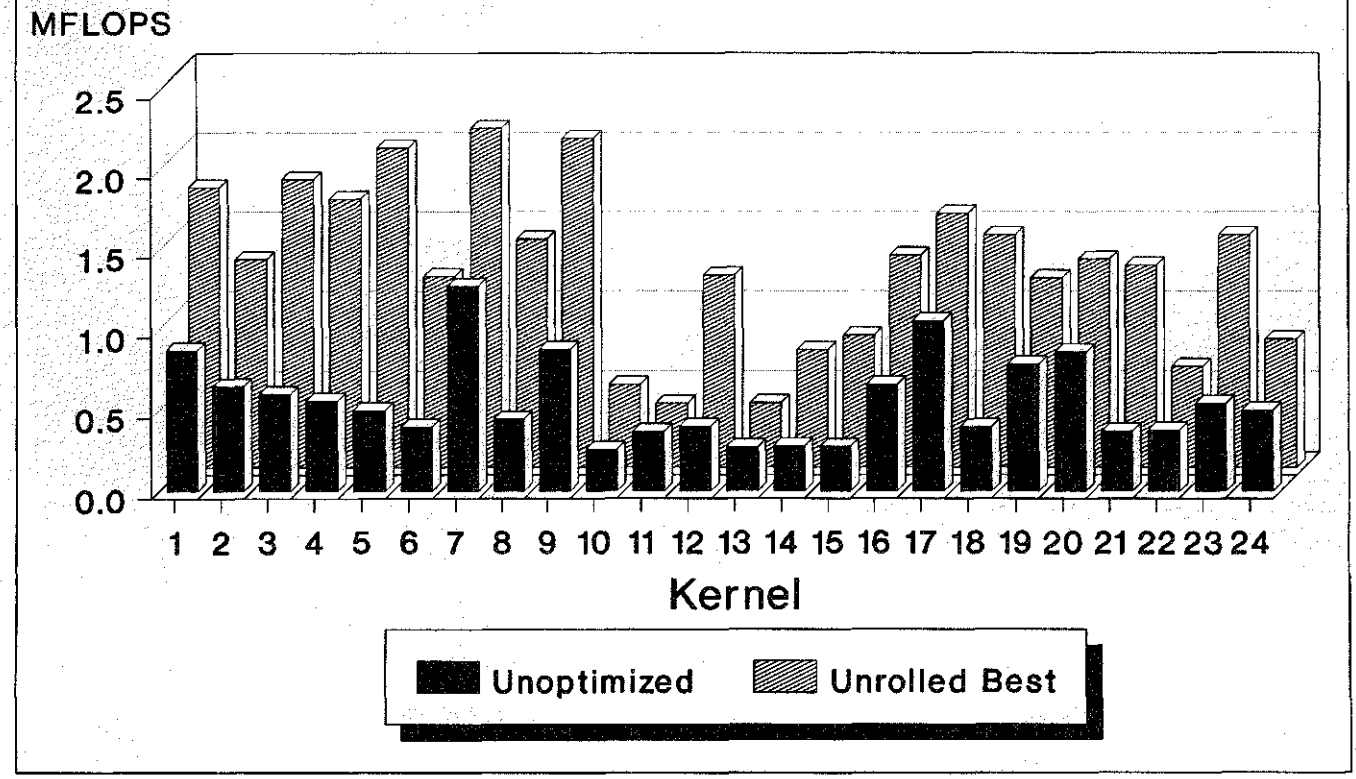


FIGURE 3: i486 optimizations for Lawrence Livermore loops



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
      DO  10  I = 1, N
         C(1,I) = A(1,1) * B(1,I) + A(1,2) * B(2,I) +
                  A(1,3) * B(3,I) + A(1,4) * B(4,I)
         C(2,I) = A(2,1) * B(1,I) + A(2,2) * B(2,I) +
                  A(2,3) * B(3,I) + A(2,4) * B(4,I)
         C(3,I) = A(3,1) * B(1,I) + A(3,2) * B(2,I) +
                  A(3,3) * B(3,I) + A(3,4) * B(4,I)
         C(4,I) = A(4,1) * B(1,I) + A(4,2) * B(2,I) +
                  A(4,3) * B(3,I) + A(4,4) * B(4,I)
10       CONTINUE
```

This restructuring will outperform the "rolled" version by about a factor of three. Reconstructing existing code is not uncommon when an application is ported from a CISC to RISC/supercomputer architecture.

THE FACTOR FACTOR

Loops get unrolled by a factor of 2, 4, 8, 16, 32, or 64. Typically, loop performance peaks at one of these values. Performance may not improve past this peak value, and may even degrade. A point of diminishing returns is reached for any of the reasons discussed above: data cache overflow, instruction cache overflow, register coloring problems, and the inability to optimize code further.

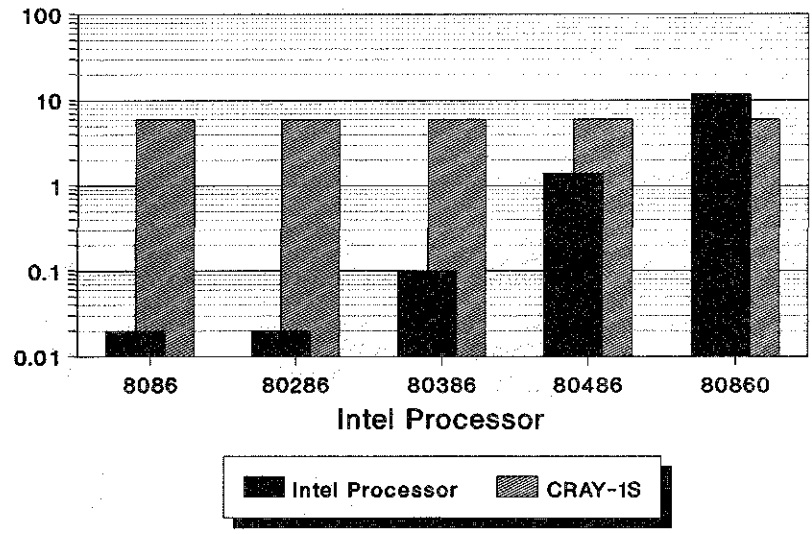
Table 2 (see p. 69) shows the 24 Livermore kernels rolled (0) and unrolled 2, 4, 8, 16, 32, and 64 times. Four patterns emerge from this data. First, all kernels show a significant improvement even after unrolling only twice. Second, for some kernels (2, 4, 5, and 8), a larger unroll factor means greater improvement. Third, for kernels 9, 10, 12, 14, 15, 17, and 22, improvement peaks at some point, and further unrolling provides no improvement. Fourth, for other kernels (1, 3, 6, 7, 11, 13, 16, 18, 19, 20, 21, 23, and 24), unrolling beyond some unroll factor produces a performance penalty. (Note that the kernels 5, 11, 16, 17, and 20 are not vectorizable by VAST.)

What is the optimal unroll factor for a given loop? That depends on which number you use. If you use the harmonic mean, unroll factor 8 wins. If the median rate is used, unroll factor 16 wins. The optimal factor also depends on the loop. As Table 2 shows, it's between 2 and 64 for a given loop; on the average, it's between 8 and 16.

Loop unrolling does require trade-offs: as the unroll factor increases, code size and compilation time also increase (see Table 1). Each programmer must evaluate these trade-offs as they relate to his or her environment.

Considering that optimal loop unrolling depends on the unroll factor, instruction

FIGURE 4: MFLOPS vs. processor performance



People are talking about us.

F77L-EM/32 & Lahey Ergo OS/386

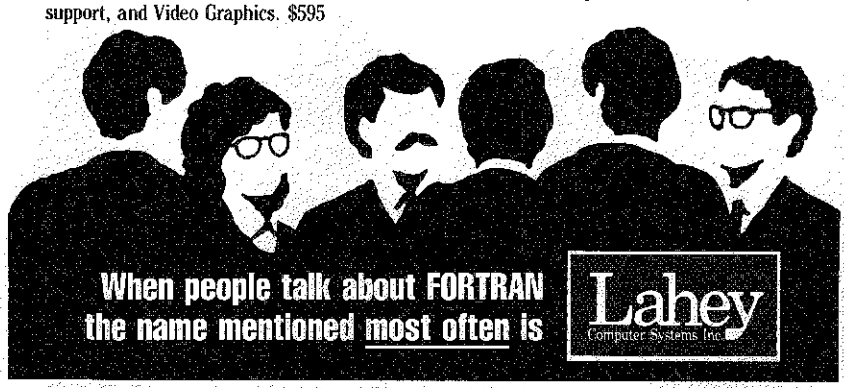
Port mainframe programs as large as 96MB to 386/486's with this 32-bit DOS-Extender compiler. The Winner of *PC Magazine's* 1988 Technical Excellence Award just got better. New Version 4.0 includes: Programming Tools, Popular Fortran 90 features, Virtual Memory Support, DESQview Support, New Documentation and Free Unlimited Runtime Licenses. F77L-EM/32 $995 OS/386 $395

F77L

The fastest real-mode compiler available. F77L can take advantage of your 386 PC by generating 32-bit instructions. New Version 5.0 includes: Fortran 90 features, Weitek support, and Video Graphics. $595

Lahey Personal Fortran 77

Version 3.0: Full ANSI 77, Editor, Debugger, Linker, Library Manager, Microsoft and Borland C interfaces. A great learning tool at an unbeatable price. $99



Contact us to discuss our products and your needs. **(800) 548-4778**
Lahey Computer Systems, Inc. P.O. Box 6091, Incline Village, NV 89450
Tel: (702) 831-2500 FAX: (702) 831-8123

FORTRAN IS OUR FORTE

Circle 49 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

cache size, number of available registers, code size, and compilation rate, can a compiler determine an optimal unroll factor "on the fly"? To do so, it would unroll a loop using each unroll factor; then it would estimate execution speed. The trade-off would be a logarithmic explosion in compile time. That's why compiler vendors such as MicroWay do not implement this "unroll evaluator." Programmer experimentation is still the best and fastest means of optimizing the unroll factor.

Comparing Figure 2 (i860 results) and Figure 3 (i486 results) (p. 70) supports a claim made above: loop unrolling has better performance gains on a RISC architecture than on a CISC architecture. On the i860, performance improves by a factor of 4.35 (7.2764 / 1.6700). On the i486, the factor is 2.8 (1.3125 / 0.4661).

My work with loop unrolling has led me to draw some firm conclusions: loop unrolling (scalar optimization) can dramatically improve performance. In certain situations, scalar optimizations can rival vector optimizations. Loop unrolling causes trade-offs in code size, compilation speed, and performance. And a point of diminishing returns is reached when loops are unrolled past some factor.

THE RACE IS ON

The Intel i860 breaks a barrier for PC computing. It is the first generally available CPU for desktop computers that outperforms a CRAY-1 (see Figure 4, p. 71). This architecture will change the course of application development and numerical experimentation in the 1990s.

PC programmers can now begin to explore applications such as weather forecasting and ocean flow analysis—experimentation that has not been available to many programmers because of the cost of traditional supercomputers.

Additionally, the i860 creates new opportunities for compiler vendors—and obstacles for some—in the arena of optimization. The i860 is fast when unoptimized, very fast when scalar-optimized, and "CRAY-like" when vector-optimized—demonstrating, again, that understanding program optimization is imperative to obtaining peak performance from a platform.

TEST ENVIRONMENT

All Lawrence Livermore loop numbers presented in this article were produced out of the box in single precision. This benchmark is distributed with NDP-FORTRAN, by MicroWay. The numbers were obtained using a 40-MHz, 32-MB i860 NumberSmasher running under Interactive Unix, hosted on a 16-MB, 33-MHz i486 NumberSmasher. To reflect a real-world program, the i860 clock was *not* turned off. Had the interrupt timer been disabled, the kernel numbers would have been inflated by 10–15%. PJ

YOU OUGHT TO BE IN
☆REPRINTS☆

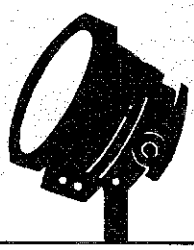
You're the picture of success.
And **PJ** reprints are a great way to showcase your four-star performance. Reprints help you:

- ☆ Enhance your trade show presentations
- ☆ Promote your products and services
- ☆ Increase your marketplace presence
- ☆ Optimize your product training sessions

PROGRAMMER'S JOURNAL offers full-color or black-and-white reprints.

PJ reprints keep you in the spotlight.

Call us today at 800-234-0386.



REFERENCES

1. Margulis, N. (1990). *i860 MicroProcessor Architecture*. *Intel*. Berkeley, CA: Osborne, McGraw-Hill.
2. Hill, M. D. (1990). "What Is Scalability?" *ACM SIG on Computer Architecture News*, **18**(4), pp. 18–21.
3. Hennessy, J. L., and Patterson, D. A. (1990). *Computer Architecture: A Quantitative Approach*. San Mateo, CA: Morgan-Kaufman, p. 50.
4. Smith, J. E., and Shlomo, W. (1990). "A Study of Scalar Compilation Techniques for Pipelined Supercomputers." *ACM Transactions on Mathematical Software*, **16**(3), pp. 223–245.
5. Fried, S. (1991). "In Search of DAXPY." Santa Clara, CA: Software 1991 Conference Proceedings.
6. Levesque, J. M., and Williamson, J. W. (1989). *A Guidebook to FORTRAN on Supercomputers*. San Diego, CA: Academic Press, Inc.
7. Smith, J. E. (1988). "Characterizing computer performance with a single number." *Communications of the ACM*, **31**(10), pp. 1202–1206.

ADDITIONAL READING

- Intel Corporation (1990). *i860 Microprocessor Programmer's Reference Manual*.

NOTE: Use of Livermore Loops is granted with the request that a copy of any results be sent to their author. Send to F. H. McMahon, Lawrence Livermore National Laboratory, P.O. Box 808, Livermore, CA 94550.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Data structure imperatives

Data structures and algorithms can boost your programming power

By Richard Riehle

I once worked for a boss who, when he had to lay people off, would call in each employee and say, "I want to talk to you about your career." Well, I want to talk to you about your career.

Graphs, trees, lists, queues, stacks, rings, maps: these are some of the common data structures required for the development of non-trivial software systems. Unfortunately, many programmers do not know how to implement such structures. I know of one programmer who fulfilled a requirement of an embedded-systems project with a linked list when he should have used a binary search tree. Because he didn't know how to write a binary search tree algorithm, the program correctly produced the desired result, but not as efficiently as it could have.

WHO NEEDS TO KNOW?

Most undergraduate computer science programs require a course in data structures. But many career programmers have developed their skill through experience rather than through formal education.

Information systems programmers are especially deficient in their knowledge of data structures and fundamental algorithms. MIS programmers often have no idea how to code a stack. Consequently, code for commercial systems tends to be both trivial and unnecessarily complex.

COBOL programmers also have been weak in this area because COBOL cannot support high-quality data structure algorithms, and coding methods for such systems tend to be derived from anecdotal rather than published algorithms. The COBOL designers had no idea that it would be useful to have a pointer data type. But without pointers, it is nearly impossible to create a dynamic structure such as a linked list. COBOL programs are thus limited largely to file-processing problems. FORTRAN 77 and BASIC also fail to provide pointer data types—a common problem with many older languages.

The example of the programmer who used a linked list instead of a binary search tree is all too familiar. In this case, it was not apparent from the requirements statement that the module should have been coded as a binary tree using an in-order algorithm. Some kind of dynamic structure was required, and the programmer did not have sufficient knowledge of data structures to select the correct solution. The program's language could have supported nearly any standard data structure.

Learning about data structures and their algorithms is definitely a career move. Without a knowledge of data structures, you're working without a complete set of tools. The preferred solution to a problem may be a doubly linked ring, but if you do not know the characteristics and algorithmic qualities of that structure, the software solution may work—and still not be ideal.

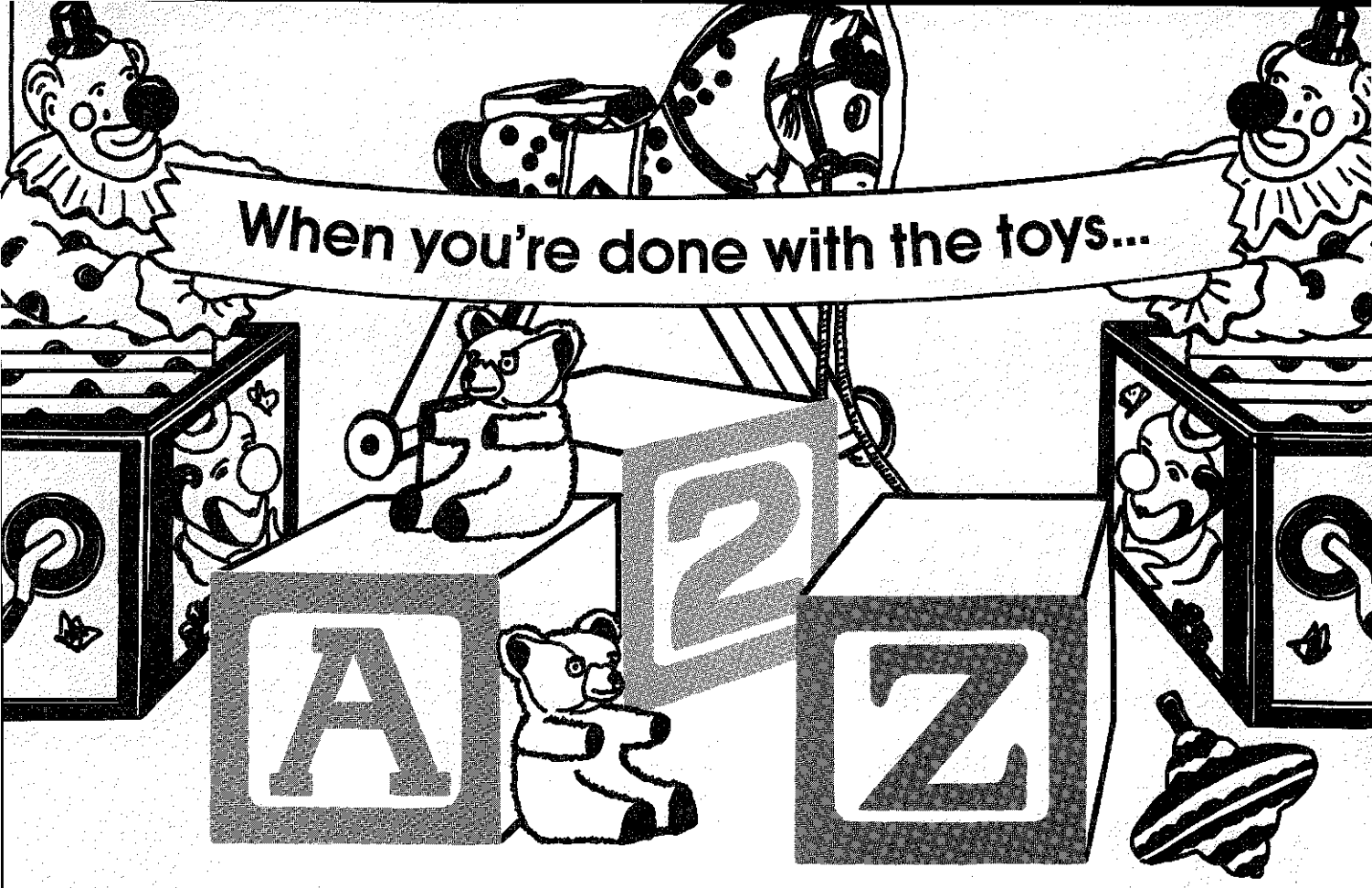
TEST YOUR EXPERTISE

Suppose you want to write a word processing package. How would you organize the lines of text and the words in each line? Remember that the user will need to insert and delete lines as well as words within lines. Moreover, you need a method to copy and move blocks of text and do rapid search and replacement. You also want to limit storage space so that the program can operate effectively using a minimum (read 640K) of memory. For example, when the user creates clipboard entries, each should occupy no additional space. Maybe you want the cursor to wrap around to the first line of text when you try to "arrow-down" beyond the last line. Would you implement this with a singly linked list, doubly linked list, doubly linked ring, binary tree, or graph? Which data structure would best represent the words for each text line? Now, include a parser to tokenize the elements of each line of text according to a predefined order. Which data structure would you use?

This problem would require, at minimum, a language that supports dynamic data types in the form of pointers. This eliminates COBOL, FORTRAN 77, BASIC, and 4GLs. The problem could be

Richard Riehle is a software engineer with AdaWorks. He can be reached at 1090-105 Tanland Drive, Palo Alto, CA 94303; (415) 858-1551.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



Janus/Ada Professional Development Systems are ready.

Janus/Ada 80386/80486 Unix

POSIX complient
ACVC 1.11 Validated
Full 32-bit compiler
Extensive Runtime Libraries
80386 Assembler and Disassembler
Janus/Ada Debugger
Interface to C
JWindows
Syntax Checker
Program Profiler

ONLY $900.00

Janus/Ada 80386 MS-DOS

ACVC 1.11 Validated
Hosted in 80386 protected mode
Targeted to the standard DOS environment
Extensive Runtime Libraries
80X86 Assembler and Disassembler
Janus/Ada Debugger including Periscope
Janus/Ada Workstation Environment
Interface to MASM
JWindows
Syntax Checker
Program Profiler

ONLY $500.00

RR Software Inc.

Circle 50 on Reader Service Card

VALIDATED Ada

THIS PRODUCT CONFORMS TO ANSI/MIL-STD-1815A AS DETERMINED BY THE AJPO UNDER ITS CURRENT TESTING PROCEDURES

P.O. Box 1512
Madison, Wisconsin 53701
U.S.A.

Telephone: 608-276-6022
Facsimile: 608-276-6009
Order line: 800-722-3248

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

solved without pointers, but not efficiently. (Of course, we could code it in assembly language.)

The issue is critical in embedded systems. In that growing domain, storage devices such as disk drives are seldom available; all information is stored in primary memory. Selecting the right data structure is crucial to effective implementation.

New languages—starting with PL/1 and moving to Pascal, C, Modula-2, Ada, and now even Fortran 90—provide the capabilities for implementing data structures. Each includes recursion as well as a powerful mechanism for indirect addressing: a pointer type. Pointers are required for C, but even C programmers are often ignorant of fundamental algorithms. Pascal programmers usually learn how to code a singly linked list in their first class, but beyond that they tend to get lost.

MOVING UP THE TREE

Learn as much as possible about structures. When you understand, for example, the three standard algorithms for traversing a binary tree, you can approach a whole new class of problems. Then you can proceed to the algorithms for balancing binary and *n*-ary trees. Next, tackle B-trees, depth- and breadth-search algorithms, and graph management and traversal.

Try learning how to code a binary tree that keeps itself balanced as new nodes are inserted and deleted, such as an Adelson-Velski and Landis (AVL) tree. Other algorithms balance an entire tree after it has been constructed. Which choice is appropriate? Should you use a recursive algorithm or simple iteration? How should you traverse the tree after it is in place? Of the three common methods for tree traversal (pre-order, in-order, and post-order), which solution calls for which method?

Then try directed and undirected graphs—useful for simulations, critical-path method diagrams, games, and network management. Few programmers can write a graph traversal program; even fewer can write a program to dynamically insert and delete nodes and edges in a graph.

OPPORTUNITY KNOCKS

These aren't academic exercises: data structures are essential for efficient implementation of the emerging applications of robotics and process control. Language parsers, expert systems applications, and complex simulations require a knowledge of data structures.

Business opportunities exist here. Grady Booch has a library of predefined data structure components for C++ and Ada. EVB Software Engineering and others have commercially available data structures in Ada. If you convince the marketplace that you have a sparkling data structure contribution that can be reused easily in real software, you can pick up extra money.

The example (Listings 1 and 2) of a doubly linked ring—written in Ada but understandable to Pascal and Modula-2 programmers—should provide interesting study material. Listing 3 (in C) demonstrates how to traverse a binary tree using an in-order algorithm. If these programs look difficult, you have your work cut out for you. PJ

PC-lint 386 Computer Language Productivity Award 1990

PC-lint presents C Bug # 604

```
#include <stdio.h>
char *hello( char *s )
    {
    char buffer[40];
    sprintf( buffer, "Hi %s", s );
    return buffer;
    }
```

Assuming that the argument string is never greater than 20 characters there is, nonetheless, a subtle bug in this C function. Can you spot it? If not, give us a call. Hint: it "works" some of the time.

PC-lint will catch this and many other C bugs. Unlike your compiler, PC-lint **looks across all modules** of your application for bugs and inconsistencies.

More than **270 error messages.** More than **90 options** for complete customization. Suppress error messages, locally or globally, by symbol name, by message number, by filename, etc. **Easy to Use** – If you know C you already know how to use PC-lint. Check for **portability** problems. Alter size of scalars. Adjust format of error messages. Automatically generate ANSI prototypes for your K&R functions.

Attn: Power users with huge programs.

PC-lint 386 uses DOS Extender Technology to access the full storage and flat model speed of your 386.

PC-lint 386 – $239
PC-lint DOS or OS/2 – $139

Mainframe & Mini Programmers

FlexeLint in shrouded source form, is available for Unix, OS-9, VAX/VMS, QNX, IBM VM/MVS, etc. Requires only K&R C to compile but supports ANSI. Pricing starts at $798. Call for details.

Gimpel Software

3207 Hogarth Lane, Collegeville, PA 19426

CALL TODAY (215) 584-4261 Or FAX (215) 584-4266

30 Day Money-back Guarantee.

PA add 6% sales tax.

PC-lint and FlexeLint are trademarks of Gimpel Software

Contact this advertiser directly

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

LISTING 1

```
-- PACKAGE SPECIFICATION for a simple doubly linked ring data structure.
-- There are many ways to implement a doubly linked ring. This Ada listing,
-- submitted by one of my undergraduate computer science students, can be
-- improved and made more reliable. Notably, using a counter to control sub-
-- programs that traverse the ring has potential dangers. It might be better
-- to have a "marked" node to serve as the starting and ending point for
-- any traversal algorithm. A good data structure on which to experiment.

with TEXT_IO;
generic.    -- define generic formal parameters
     type DLRing_Type is private;
     with function  EQUAL (LEFT, RIGHT : DLRing_Type) return BOOLEAN;

package DLRing is
   type DIRECTION_TYPE     is (FORWARD, BACKWARD); -- rotation
   type FINGER_TYPE        is limited private;

   procedure ROTATE   (DIRECTION    : in    DIRECTION_TYPE;
                       FINGER       : in out FINGER_TYPE);
   procedure INSERT   (ITEM         : in     DLRing_Type;
                       FINGER       : in out FINGER_TYPE);
   procedure REPLACE  (ITEM         : in     DLRing_Type;
                       FINGER       : in out FINGER_TYPE);
   procedure DELETE   (FINGER       : in out FINGER_TYPE;
                       EMPTY_ERROR  :    out BOOLEAN);
   procedure PEEK     (ITEM         :    out DLRing_Type;
                       FINGER       : in out FINGER_TYPE;
                       EMPTY_ERROR  :    out BOOLEAN);
   procedure SEARCH   (FINGER       : in out FINGER_TYPE;
                       ITEM         : in     DLRing_Type;
                       FOUND        :    out BOOLEAN);
   function  Is_Count (Finger : Finger_Type) return Natural;

private
         type CELL;                      -- an incomplete type
         type POINTER is access CELL;    -- set up a pointer type
         type FINGER_TYPE is  record     -- keeps track of where
                    PTR  : POINTER;      -- we are in the ring
                    CNTR : NATURAL := 0;
               end record;
end DLRing;
```

LISTING 2

```
-- PACKAGE BODY  Doubly Linked Ring
-- There would be some benefit in adding storage management capability,
-- such as a free list handler, to this package. Also, a storage_error
-- exception would enhance the reliability of the Insert procedure.

package body DLRing is
       ZERO : constant INTEGER := 0;  -- named constant
       type CELL is record
                  ITEM        : DLRing_Type;
                  PREV_CELL   : POINTER;
                  NEXT_CELL   : POINTER;
            end record;

   function Is_Count (Finger : Finger_Type) return Natural is
   begin
        return Finger.Cntr;
   end Is_Count;

   procedure ROTATE   (DIRECTION : in DIRECTION_TYPE;
                       FINGER    : in out FINGER_TYPE) is
   begin  -- ROTATE
      if (DIRECTION = FORWARD) then
```

continued

Memory Problems Solved!

MS-DOS 'C' AND ASSEMBLER PROGRAMMERS - BREAK THE 640K LIMIT!

XMEM A complete virtual memory system! — $149

- Manages expanded, extended, and can swap to disk.
- Eliminates memory fragmentation problems.
- Optionally emulates Windows & Macintosh memory managers.

EMPTY SHELL Run any program from your program! — $169

- Shell to any program, leaves less than 10K resident!
- Saves and restores screen - text or graphics!
- Swaps your program to expanded memory, extended memory or disk.

MEMORY TOOLKIT Easy access to all memory! — $99

- Read and write expanded or extended memory with simple file like calls.
- Create multi-megabyte arrays in expanded, extended, or on disk.
- Complete set of calls to EMS and XMS managers.

To Order Call:
1•800•755•7973
Visa / MasterCard / C.O.D.
International: (609) 587-7973
Fax: (609) 587-9412

Kandu Inc.
"We Can Do What Others Can't"

ALL OUR PRODUCTS FEATURE: • Source code • Tested with Microsoft, Turbo, & Zortech C • 30 Day Money-back Guarantee • No Royalties! • 100% Assembler • Free technical support

Circle 51 on Reader Service Card

Program FASTER with ...

CC-RIDER

or the new

C++ RIDER

POP-UP SOURCE BROWSERS FOR ANY EDITOR!

CROSS-REF Editing!
Instantly pop-up your symbol definitions with a hypertext hot-key inside any editor! The Standard Edition has Source View, Edit and Paste commands. And the powerful *Professional Edition* even lets you walk through all uses of a symbol in your code as you edit!

Source Analysis!
Build a database of symbol information with our powerful ***source code analyzer*** which generates a wealth of useful output, like fully commented static and global function prototype #include files and a detailed log file showing WHERE and HOW all your symbols are used. And it uses expanded memory for ***lightning fast*** source code analysis!

QuickHelp, too!
CCSYM can also create a ***help database for your program*** which is compatible with Microsoft's help system, *QuickHelp*. All symbols in your application are accessible from QuickHelp's menus, with ***documentation extracted automatically from your original source code.***

Works with ANSI/Microsoft C, plus AT&T, Glockenspiel, Zortech, Borland C++. Supports Microsoft C v6 extensions.

FREE DEMO !

Standard	*Professional*
$89	***$279***
DOS or OS/2 C or C++	DOS & OS/2 C & C++

Western Wares
(303) 327-4898
Box C Norwood, CO 81423

Circle 52 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
        FINGER.PTR := FINGER.PTR.NEXT_CELL;
    else
        FINGER.PTR := FINGER.PTR.PREV_CELL;
    end if;
exception
     when CONSTRAINT_ERROR =>
         Text_IO.Put_Line("Constraint Error during Rotate ");
end ROTATE;

procedure INSERT   (ITEM      : in     DLRing_Type;
                    FINGER    : in out FINGER_TYPE) is
          ForWard_Walker  : POINTER;
          BackWard_Walker : POINTER;
begin  -- INSERT
    if FINGER.CNTR = ZERO then
       FINGER.PTR  := new CELL'(ITEM,null, null);
       FINGER.PTR.NEXT_CELL := FINGER.PTR
       FINGER.PTR.PREV_CELL := FINGER.PTR;
    else
       BackWard_Walker  := FINGER.PTR.PREV_CELL;
       ForWard_Walker   := FINGER.PTR;
       FINGER.PTR       := new CELL'(ITEM, BackWard_Walker,
                                     ForWard_Walker);
       BackWard_Walker.NEXT_CELL := FINGER.PTR;
       ForWard_Walker.PREV_CELL  := FINGER.PTR;
    end if;

     FINGER.CNTR := FINGER.CNTR + 1;  -- could be dangerous unless every
                  -- subprogram handles the count in consistent manner.
exception
     when others =>
        Text_IO.Put_Line("Some error during Insert ");
end INSERT;

procedure REPLACE  (ITEM      : in     DLRing_Type;
                    FINGER    : in out FINGER_TYPE) is
begin --  REPLACE
          FINGER.PTR.ITEM  := ITEM;
end REPLACE;

procedure DELETE   (FINGER       : in out FINGER_TYPE;
                    EMPTY_ERROR  :    out BOOLEAN)   is
         BackWard_Walker, ForWard_Walker  : POINTER;
begin
     EMPTY_ERROR := FALSE;
     if FINGER.CNTR  = ZERO  then
        EMPTY_ERROR  := TRUE;
     elsif
        FINGER.CNTR = 1 then
        FINGER.PTR  := null;
        FINGER.CNTR := ZERO;
     else
        BackWard_Walker           := FINGER.PTR.PREV_CELL;
        ForWard_Walker            := FINGER.PTR.NEXT_CELL;
        BackWard_Walker.NEXT_CELL := ForWard_Walker;
        ForWard_Walker.PREV_CELL  := BackWard_Walker;
        FINGER.CNTR               := FINGER.CNTR - 1;
        FINGER.PTR                := ForWard_Walker;
     end if;
end DELETE;

procedure PEEK     (ITEM        :    out DLRing_Type;
                    FINGER      : in out FINGER_TYPE;
                    EMPTY_ERROR :    out BOOLEAN)   is
begin
     if (FINGER.CNTR  /=  ZERO)  then
         ITEM        := FINGER.PTR.ITEM;
         EMPTY_ERROR := FALSE;
     else
         EMPTY_ERROR  := TRUE;
     end if;
exception
     when Constraint_Error =>
        Text_IO.Put_Line("Constraint Error in Peek ");
end PEEK;

procedure SEARCH   (FINGER       : in out FINGER_TYPE;
                    ITEM         : in     DLRing_Type;
                    FOUND        :    out BOOLEAN)   is
          NUMBER : INTEGER := FINGER.CNTR;
begin -- SEARCH
      for I in 1 .. NUMBER  loop
          if (EQUAL(FINGER.PTR.ITEM, ITEM)) then
              FOUND := TRUE;
              return;
          else
              ROTATE(FORWARD, FINGER);
          end if;
      end loop;
      FOUND := FALSE;
      ROTATE(FORWARD, FINGER);
   exception
        when CONSTRAINT_ERROR =>
            Text_IO.Put_Line("Constraint error in Rotate ");
   end SEARCH;
end DLRing;
```

LISTING 3

```
/* A simple binary search tree implementation. A tree is a special case of
the more general structure, the directed graph. A complete implementation
would require more features, including memory management, error handling,
and utility functions. To use a binary search tree in an embedded system ap-
plication, it would be necessary to include a "free list" handling func-
tion. This algorithm does not include a capability to keep the nodes of the
tree balanced. Consequently, a realistic implementation would also include
some balancing algorithm such as the AVL method. */

#include <stdio.h>
typedef struct NODE {
        int    data;
        struct node  *left;
        struct node  *right};
#define TREESIZE 100
#define NUL 0
void filltree(struct *NODE), inorder(struct *NODE), postorder(struct *NODE);
int getnumber();

void main() {
{
    struct NODE numbers[TREESIZE];
    filltree(numbers);     /* insert items into tree */
    inorder(number);       /* display tree using inorder traversal */
    postorder(number);     /* display tree using postorder traversal */
}
void filltree (struct NODE *tree) {
    int index = 0;
    int num;
    while ((num=getnumber()) != -1) {
        tree[index].data  = num;
        tree[index].left  = tree[index].right = NUL;
        if (index > 0) insert (tree, &tree[index]);
        index++;
    }
}
int getnumber() {
    int num;
    printf("\n Enter Positive Integer (-1 to end): ");
    scanf("%d", &num);
    return(num);
}
insert (struct NODE *tree, struct NODE *newnode) {
   if (newnode -> data < tree -> data) {
       if (tree -> left == NUL)
          tree -> left = newnode;
       else
          insert (tree -> left, newnode);
   }
   else {
       if (tree -> right == NUL)
          tree -> right = newnode;
       else
          insert (tree -> right, newnode);
   }
 void inorder (struct NODE *tree)  {    /* notice the placement of printf */
   if (tree -> left != NUL)
       inorder (tree -> left);        /* call left child again      */
   printf ("%d\n", tree -> data);     /* visit the node             */
   if (tree -> right != NUL)
       inorder (tree -> right);       /* call right child again     */
}
void postorder (struct NODE *tree)  {  /* notice the placement of printf */
   if (tree -> left != NUL)        /* call left child again       */
       postorder (tree -> left);
   if (tree -> right != NUL)
       postorder (tree -> right);  /* call right child again      */
   printf ("%d\n", tree -> data);  /* visit the node              */
 }
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Text filters in BASIC

User-friendly data transformation without pain or strain

By Murray L. Lesser

Although its inventors may not have intended as much, BASIC has become an ideal programming language for writing text filters. Unlike other compiled languages, BASIC offers the combination of automatic variable-length string management, inherent complex string operations, automatic management of input/output file buffers (with sizes specified in the file OPEN statements), and automatic error detection. Also, not by coincidence, BASIC sequential files are identical to DOS ASCII (text) files. The programmer need not worry about handling any of these necessary nuisances explicitly, and the language support will take care of all the exception cases. The only requirement for writing a text filter, beyond implementing the filter algorithm correctly, is to have a pleasing user interface that includes enough "progress report" display output to identify the source of any BASIC error reports if the program terminates unexpectedly.

To illustrate, I have made a small filter to remove the effects of a subtle, date-sensitive bug that was introduced with Microsoft's MASM version 5.1. This bug infects the first line of each page header in the output listing (LST) file, but only on 81 days of each year. If the month and day can each be represented by a single digit (e.g., 4/1/92), the bug inserts an extra <LF> symbol in front of the DOS ASCII EOL symbol (the <CR><LF> pair) that marks the end of that header line.

For historic (compatibility) reasons, Microsoft BASIC parses the <LF><CR> pair as a line-continuation symbol. Thus, when a listing utility written in BASIC reads a bug-infested listing file, the two header lines on each page are concatenated into a single text line containing the embedded three-symbol sequence <LF><CR><LF>. On LPRINTed output, these symbols insert two blank lines between the two header lines. (A listing utility written in C would not concatenate the two lines but would insert one blank line on output.)

DEFENSIVE, BUT FRIENDLY

In computerese, a *filter* is a program with one input and one output that performs a useful transformation on data as it passes through. Some authorities (including the authors of the IBM *Disk Operating System* manual) have complicated this simple definition by insisting that filters should be programmed to read from standard input (the keyboard) and write to standard output (the display screen), thereby requiring the hapless user to specify redirection in each direction every time the filter is run.

My BASIC filters read an input file that has been specified directly as a command-line parameter. This allows a bit of defensive programming to be included, which would not be feasible if redirection were required. Because STRIPIT.BAS (Listing 1) is used only with MASM listing files, it won't even try to read files having any other extension. This means that the input file extension can be omitted from the calling command line. For further user friendliness, most of my text filters do not require any output file name to be specified; they write to a file whose name is derived from that of the input file. If no runtime errors have occurred, STRIPIT's output file has the same name as, and replaces, the input file.

Because BASIC's LINE INPUT statement usually removes the EOL symbols on input, a <LF> symbol will be found only if the two header text files have been concatenated. STRIPIT looks for the first <LF> in the line and removes it if found. The remaining embedded EOL symbol pair causes the concatenated header lines to appear in the output file as two separated text lines.

STRIPIT looks for this extraneous <LF> symbol only in the first header line on each page, which is identified by a <FF> symbol as the first character in the line. For performance reasons, the test is made by the compound conditional statement

---

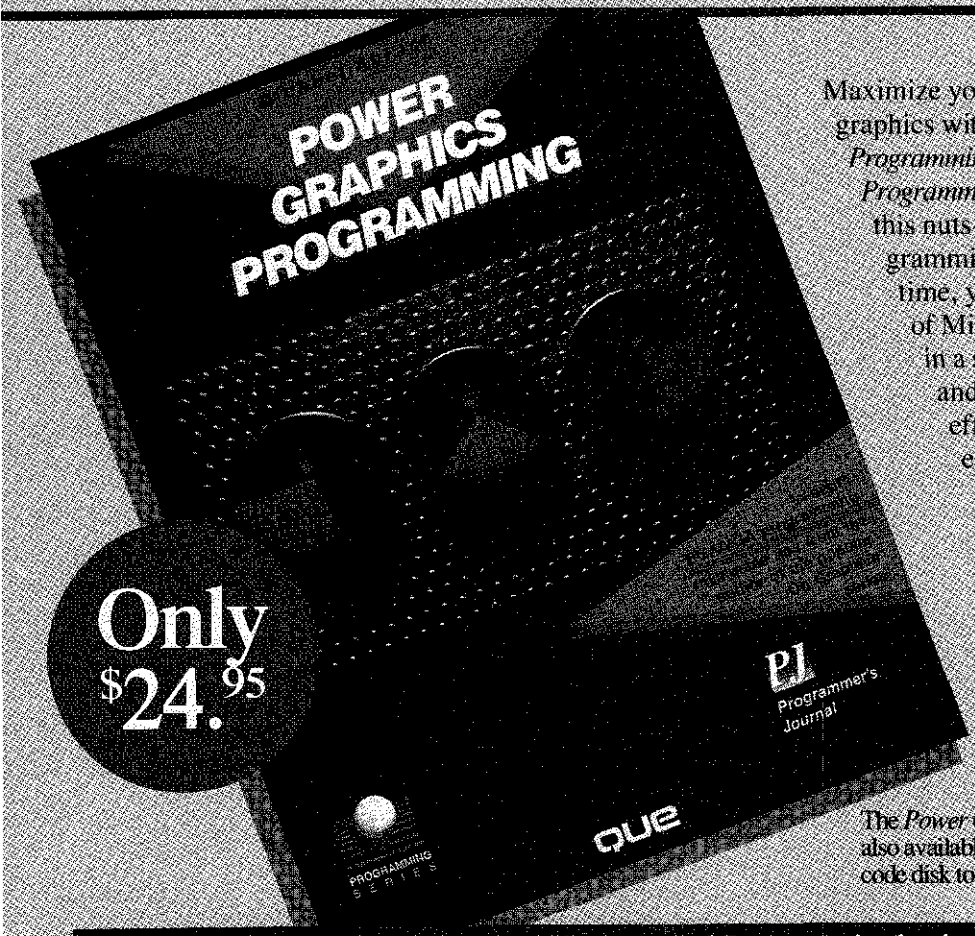
Murray L. Lesser is a retired IBM engineer. He has written three books about enhancing various Microsoft BASIC compilers with assembled modules. The most recent, Advanced QuickBASIC 4.0, *was published by Bantam.*

ILLUSTRATION BY JIM CARPENTER

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The complete *Programmer's Journal* "On Graphics" series—assembled in one volume for the very first time!

Graphic Results!



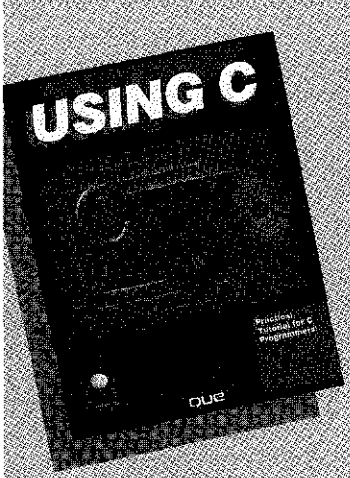
Only $24.95

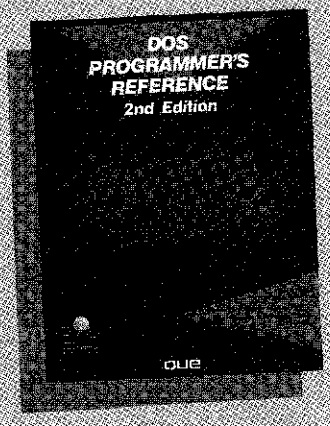
Maximize your EGA/VGA programming graphics with Que's *Power Graphics Programming*! Que has teamed up with *Programmer's Journal* to bring you this nuts-and-bolts graphics programming guide. Now, for the first time, you can have the expertise of Michael Abrash consolidated in a single volume. His insights and approach to efficient, effective programming have earned him high acclaim for his "On Graphics" series. If you are seeking that elusive graphic edge, you can't be without *Power Graphics Programming*!

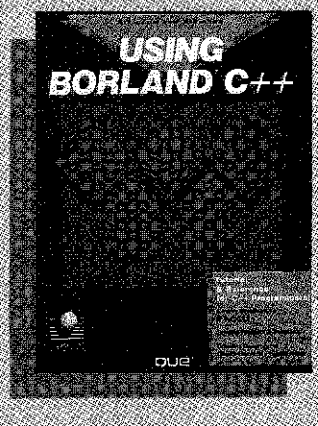
Power Graphics Programming is only available from *Programmer's Journal*. Call 800-234-0386.

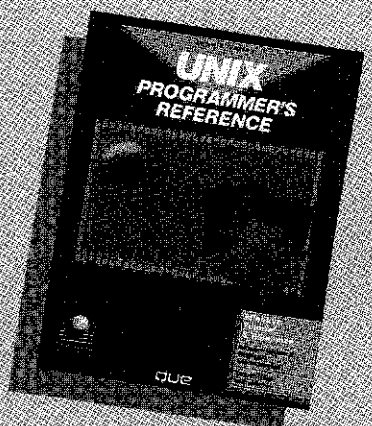
The *Power Graphics Programming* code disk is also available for $14.95. Order your book and code disk today!

Look for these other practical Programming books from Que









Available in better bookstores and computer stores everywhere.
Or call 800-428-5331, ext. PJOL

que®

PRODUCED BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

ADVERTISER INDEX

Reader Service	Advertiser Name	Page Number
45	Austin Code Works, The	59
25	Autumn Hill Software	26
110	Az-Tech Software, Inc.	93
12	Barton Creek Software	12
8	Blaise Computing, Inc.	10
60	Borland International	C4
115	Burton Systems Software	93
*	C.C. Software	93
46	Catenary Systems	60
*	Cater Software	95
*	C.D.P. Consultants	93
133	Cnapse	94
40	Data Techniques, Inc.	47
131	Discus Distribution Service	95
31	Disk Software, Inc.	33
111	DSE Software Publishing	93
130	EMS	95
112	EZX	93
10	Genus Microprogramming, Inc.	12
*	Gimpel Software	75
17	Greenleaf Software, Inc.	17
39	GUI Computer	47
117	Impulse Engineering	93
129	Information Modes	95
113	Interstitial Software Systems	93
13	Island Systems	14
37	Kadak Products, LTD	42
51	Kandu, Inc.	76
20	Knowledge Dynamics Corp.	20
54	Laboratory Microsystems, Inc.	82
49	Lahey Computer Systems, Inc.	71
29	Magee Enterprises, Inc.	32
59	Meridian Software Systems	C3
9	MetaWare Incorporated	11
14	Micro Security Systems	15
122	MicroQuill	94
*	Microsoft Corporation	1
*	Microsoft Corporation	9
134	Microtools	94
43	Mix Software	56
22	Modular Software Systems	21
61	MultiScope, Inc.	13
33	Nanosoft	38
119	Network Intergrated Services	93
5	Nu-Mega Technologies	4
*	Olivadoti Associates, Inc.	93
126	Opt-Tech	95
15	Opus Software	15
30	PC Tech, Inc.	33
58	PC Techniques	90
35	Periscope Company, The	39
23	Petrasoft	21
18	Programmer's Corner, The	19
*	Programmer's Journal	16
27	ProGraphx Toolbox	29
123	Quaid Software Limited	94
36	Qualitas, Inc.	41
34	Quantasm Corporation	38
1	Quarterdeck Office Systems	C2
4	Quarterdeck Office Systems	3
*	Que Corporation	79
120	Quibus Enterprises	94
44	Rational Systems, Inc.	57
6	RJSwantek & Associates	7
50	RR Software, Inc.	74
41	RSS Engineering	52
128	Ryle Design	95
11	S.L.A.M. Software	14
135	Selective Software	94
16	Sequiter Software, Inc.	31
24	SLR Systems	23
47	Software Blacksmith	60
57	Software Development '91	89
118	Software Mart	93
55	Software Success	82
124	Software Toolz, Inc.	94
32	Softway, Inc.	35
28	South Mountain Software	2
121	Strategic Software Designs Inc.	94
21	Sub Systems	20
125	Sutrasoft	94
48	Tool Makers, The	69
19	Tools & Techniques, Inc.	19
127	TSR Systems	95
38	TurboPower Software	43
132	Upper Deck Systems	95
56	Utilis, Inc.	83
42	V Communications	53
116	Wenham Software Co.	93
52	Western Wares	76
26	Willies' Computer Software	29

When contacting any of the above advertisers, please tell them you saw their ad in ***Programmer's Journal*** and that you appreciate their continuing support of this magazine. The advertisers marked with an asterisk (*) prefer to be contacted directly. ***PJ***'s Advertiser Index is provided as a free service, and ***Programmer's Journal*** cannot be held responsible for omissions or errors. ***Programmer's Journal*** is not responsible for errors in advertisements.

EDITORIAL INDEX

Reader Service	Vendor Name	Page Number
182	ASCI	91
196	Barton Creek Software	92
174	Blaise Computing Inc.	88
190	Butler Computer Systems	92
187	Cooper Software	91
166	Edsun Laboratories, Inc.	35
171	FTP Software, Inc.	88
189	GFA Software Technology	92
173	Gimpel Software	88
176	Greenleaf Software Inc.	88
191	Greenleaf Software Inc.	92
188	GUI Computer	91
194	Island Systems	92
177	Magna Carta Software, Inc.	88
172	Mark Williams Company	88
184	Microsoft Corporation	91
186	Microsoft Corporation	91
199	Microtools	92
185	MultiScope, Inc.	91
183	NetManage Inc.	91
167	Network Products Corporation	42
168	Novell	42
170	Nu-Mega Technologies	88
169	Periscope Company, The	88
192	Sequiter Software Inc.	92
178	SilverWare	88
198	The Software Annex	92
179	Solid Software	88
197	Solutions Systems	92
175	South Mountain Software	88
193	TurboPower Software	92
180	Willies' Computer Software Co.	88
195	Zircel Software	92

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

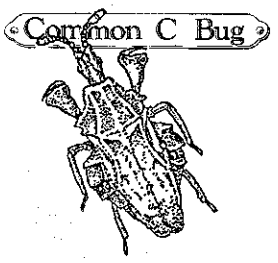
PJ ISSUE SPECIAL!

SAVE $2.00 ea.

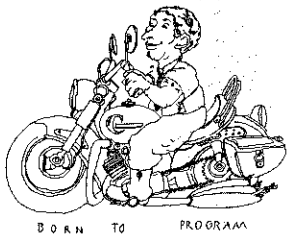
Multi-color imprints on the highest quality 100% cotton T-shirts.

SIZES: S, M, L, XL, XXL

$14.95 each — includes U.S. shipping



COMMON C BUG

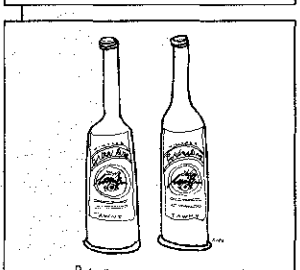


BORN TO PROGRAM

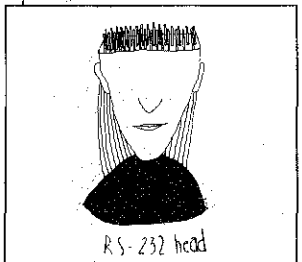
ALL CODE DISKS ONLY $8.00 EACH!

For a limited time, you can save 20% on each code disk you order. Save yourself time and money — order the disks you need today!

Free U.S. shipping. Add $2.50 shipping per disk to Canada or Mexico, $5.00 shipping per disk to any other foreign destination. Offer expires August 31,1991. Not valid for PGP disk.



PARALLEL PORTS



RS-232 HEAD

BUGS RULE 4 color imprint on a white shirt

$7.00 each PREPAID ONLY

A few of the Back Issues currently available

Vol. 8.2: GRAPHICS: Faster circles for the VGA; the TIGA interface from TI; the EGA palette; 800x600x16 Super-VGA mode programming; dynamic data exchange for communications; Manipulating large data; fast file I/O in QuickBASIC; industry news; 1989 article index.

Vol. 8.3: OPERATING ENVIRONMENTS: Undocumented DOS shell; Graphics programming in windows; Mousing around: Messages in PM; Accessing high memory; ASM: Advanced segmentation; Ellipses on the VGA; Up OOP alley in Pascal.

Vol. 8.4: PROGRAMMING IN THE PROTECTED MODE: Faster VGA ellipses and more; Running unprotected virtual 8086 mode; Converting to 386 protected mode; OS/2 2.0 memory model for the 386; Fortran for the 386; Using C++ features in C; Different times compensating for system clock idiosyncracies; ASM: Errors and stream I/O; Cobol-85 nested; Deliberately breaking the rules.

Vol. 8.5: OBJECT_ORIENTED PROGRAMMING: More colors on the VGA; A virtual memory system in C/C++; Pitfalls in Turbo Pascal OOP; Writing applications that travel; Late binding; Tools for 386 DOS; Windowing text with X; Cache memories; Breaking the rules again; SPARCs plot world takeover.

Vol. 8.6: THE DATABASE CHALLENGE: VGA color paging; The database dilemma; Networking databases; B-tree indexing; Virtual memory paging; Writing an ANSI C preprocessor; Peak performance: On to the 80486; Calendar conversions; Software components; Optimization strategies; Pink noise; Oh say can you C?

Vol. 9.1: GRAPHICS AND WINDOWING: VGA color cycling; Digital typography; Windowing for two; GUI madness; Text buffers in windows; What's on the menu; 386 DOS meets ANSI C; Optimania; Real-mode virtual memory; Enhanced keyboards; An open look at Motif; Negotiating tricks and traps; For a few good books; Goliath beats David—As usual.

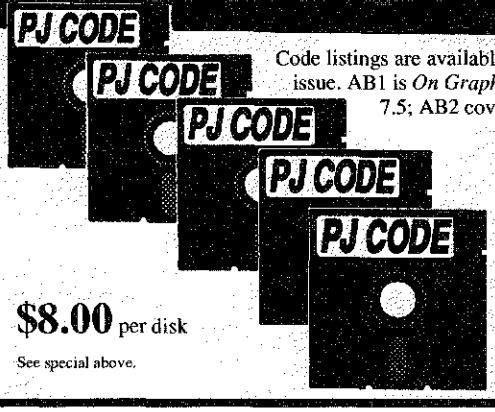
Vol. 9.2: LANGUAGES: Windows 3 input hooks; C compilers for Windows; Fortran 90 is coming!; Pragma interface; String handling; Porting to the Mac; Disassembler roundup; Exploring XMS; Using EMS for fast file I/O; Around the bend again; Pink noise; Language lore; Effective marketing presentations; COBOL revisited; Language parsers and Fortran tools; Changes in the fast lane.

Vol. 9.3: HARDWARE FOR DEVELOPERS: Pixel panning; Hardware-assisted debuggers; Object-oriented SCSI programming; Numeric exception handling; Virtual realities; BASIC libraries; BIOS compression; Hashing functions; Computer stores and popcorn kernels; Programmer productivity; Language translation and 386 debugging; Hardware help; Hardware decisions.

10% discount on any order of 10 or more T-Shirts, Disks or Back Issues.

All mail orders must be accompanied by check or Money Order -- PREPAID ONLY

Use the order form on the card opposite.
Order by phone with your VISA/MC.
CALL TOLL FREE
800/234-0386 ext 53



$8.00 per disk

See special above.

Code listings are available from volume 5.5 to the current issue. AB1 is *On Graphics* series code from issue 5.1 to 7.5; AB2 covers issue 7.6 to the current issue.

Diskettes are available for the following code:

AB1	6.4	7.5	8.6
AB2	6.5	7.6	9.1
5.5	6.6	8.1	9.2
5.6	7.1	8.2	9.3
6.1	7.2	8.3	9.4
6.2	7.3	8.4	
6.3	7.4	8.5	

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
IF LEN(TEXT) <> 0 AND ASC(TEXT) = 12 THEN
```

rather than by the logically equivalent

```
IF LEFT$(TEXT,1) = CHR$(12) THEN
```

The LEN test prevents an attempt by the runtime library code to evaluate the ASC function of a null string, which would force an "Illegal function call" runtime error.

PARDON MY BASIC

This compound conditional statement will not work with BASIC compilers prior to BC 7. Under the BASIC precedence rules in effect before BC 7, *all* relational operators in a statement are evaluated before *any* logical ones, thus forcing the ASC("") runtime error before the preceding AND operator is evaluated. Starting with BC 7, the relational expressions in a compound conditional statement are evaluated left to right only until the final logical outcome is determined. (Unfortunately, this change was not documented in Microsoft's manual, so it is an unadvertised BASIC 7 special.) If you are using the QuickBASIC compiler (BC 4.5), you must change the compound statement to a pair of nested **IF** blocks, as shown in the comments in Listing 1. When compiled with BC 7.1, either the single compound conditional statement or the nested **IF**-block statements will produce the same executable code.

Most PRINT statements in STRIPIT are to display a progress report as the filter operates. Sample output might read:

```
Stripping extraneous <LF>s from FILER.LST
    Renaming FILER.LST to $$$.$$$
    Filtering $$$.$$$ to produce new FILER.LST
        Filtering line 535
    Deleting $$$.$$$
STRIPIT ran to completion.
```

If STRIPIT is unable to complete, the source of the error exit can be determined from the text of the last progress report line and from the text of BASIC's error message. For example, if the input FILER.LST file has the **read only** attribute bit set, the last progress report in the display would be "Deleting $$$.$$$," and the BASIC error message terminating the program would be "Path/File Access Error" (the file would be renamed without changing the attribute bit). Both the renamed input file and the output file (FILER.LST) would be shown in the input directory. If the input file were bug free, the two files would be identical.

Once you have written a text filter in BASIC, writing others is mainly a matter of editing the source code. Most of the executable code is "boilerplate," requiring only minor modification for different input and output file names. The major change is to the filter algorithm—typically, fewer than a third of the executable statements. The BASIC support system does the hard part for you. PJ

Total control with LMI FORTH™

For Programming Professionals: an expanding family of compatible, high-performance compilers for microcomputers

For Development:
Interactive Forth-83 Interpreter/Compilers for MS-DOS, OS/2, and the 80386

- 16-bit and 32-bit implementations
- Full screen editor and assembler
- Uses standard operating system files
- 500 page manual written in plain English
- Support for graphics,floating point, native code generation

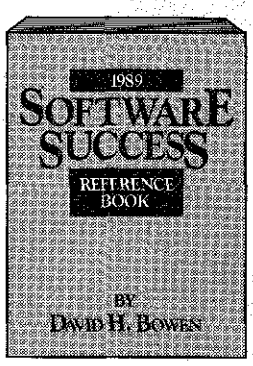
For Applications: Forth-83 Metacompiler

- Unique table-driven multi-pass Forth compiler
- Compiles compact ROMable or disk-based applications
- Excellent error handling
- Produces headerless code, compiles from intermediate states, and performs conditional compilation
- Cross-compiles to 8080, Z-80, 8086, 68000, 6502, 8051, 8096, 1802, 6303, 6809, 68HC11,34010, V25, RTX-2000
- No license fee or royalty for compiled applications

Laboratory Microsystems Incorporated
Post Office Box 10430, Marina del Rey, CA 90295
Phone Credit Card Orders to: (213) 306-7412
FAX: (213) 301-0761

Circle 54 on Reader Service Card

THE ANSWER BOOK FOR YOUR SOFTWARE MARKETING PROBLEMS!



SOFTWARE SUCCESS REFERENCE BOOK 1989 by David H. Bowen

I bet at least once today, you've already been faced with a tough decision in some area of your business. Was it Promotion? Lead Generation? Sales? Pricing? A Legal or Management issue? Don't you wish you could pull a book off your shelf and, within seconds have a solution to your problem? Does such a book exist? YOU BET IT DOES!

It's THE SOFTWARE SUCCESS REFERENCE BOOK, a 262-page guide, organized by topic to provide you with FAST HELP in solving your TOUGH PROBLEMS. Compiled from a full year of *SOFTWARE SUCCESS* — the "whole business" newsletter for software CEOs — this indispensable answer book will save you hours of needless worry over making the right decisions for your software company.

Send your check for $50; or call or FAX with your credit card payment (VISA/MC/AEX).
BY PREPAYING, YOU'LL RECEIVE A FREE 3-MONTH SUBSCRIPTION to *SOFTWARE SUCCESS* — a $57 value!

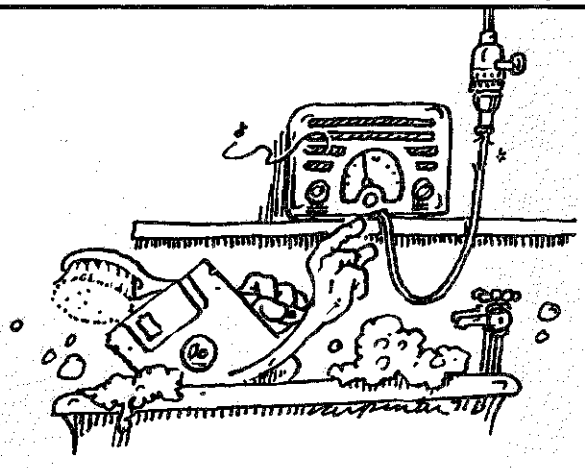
SOFTWARE SUCCESS
P.O. Box 9006
San Jose, CA 95157
Ph: (408) 446-2504
Fax: (408) 255-1098

Circle 55 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

LISTING 1

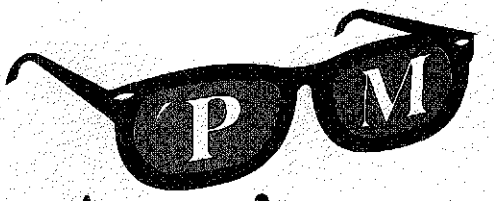
```
                              STRIPIT.BAS
'     STRIPIT deletes any extraneous <LF> characters inserted into the
' page heading of a MASM listing (LST) file by the bug in MASM 5.10.
' (The erroneous LST files occur only on dates in which both the month
' and the day are given by single digits.)  The original file will be
' replaced by the "stripped" file.
'                    Written by M. L. Lesser, 2/14/91
'            Compiled with Microsoft BC v 7.1, switch "/O"
'   Linked to stub files NOFLTIN, NOCOM, and NOLPT, switches "E/NOE"

   DEFSTR F, T: DEFINT I, L
   LET FILE.OUT = COMMAND$              '<filespec> from command line
   IF LEN(FILE.OUT) = 0 THEN
       PRINT: PRINT "No input <filespec> on command line."
       PRINT "    Usage:  Call with STRIPIT <filespec>"
       PRINT "              where <filespec> is a MASM .LST file."
       END
   END IF
' Force FILE.OUT to have .LST extension:
   LET I = INSTR(FILE.OUT,".")
   IF I <> 0 THEN LET FILE.OUT = LEFT$(FILE.OUT,I)
   LET FILE.OUT = FILE.OUT + ".LST"
   COLOR 2,0                                        'Dealer's choice
   PRINT "Stripping extraneous <LF>s from "; FILE.OUT
   PRINT TAB(5) "Renaming "; FILE.OUT; " to $$$.$$$"
   NAME FILE.OUT AS "$$$.$$$"
   PRINT TAB(5) "Filtering $$$.$$$ to produce new "; FILE.OUT
   OPEN "$$$.$$$" FOR INPUT AS #1 LEN=5120       'Use 5KB buffer
   OPEN FILE.OUT FOR OUTPUT AS #2 LEN=5120
   WHILE NOT EOF(1)
       LINE INPUT #1, TEXT
       IF LEN(TEXT) <> 0 AND ASC(TEXT) = 12 THEN    'Header-line check
' Note:  To prevent an "Illegal function call" runtime error if compiled
'      with BC version 4.5, comment out the statement above this note;
'      use the three commented-out (nested IF block) statements below
         IF LEN(TEXT) <> 0 THEN
           IF ASC(TEXT) = 12 THEN
             LET I = INSTR(TEXT,CHR$(10))          '  (marked by <FF>)
             IF I <> 0 THEN                        '  for extraneous <LF>
                 LET TEXT = LEFT$(TEXT,I-1) + MID$(TEXT,I+1)
                 LET LINE.NO = LINE.NO + 1
             END IF
           END IF
       END IF
       LET LINE.NO = LINE.NO + 1
       LOCATE ,1                                    'Display filter progress
       PRINT TAB(10) "Filtering line" LINE.NO;  ' by overprinting
       PRINT #2, TEXT
   WEND
   PRINT
   CLOSE
   PRINT TAB(5) "Deleting $$$.$$$"
   KILL "$$$.$$$"
   PRINT "STRIPIT ran to completion."
   END
```



Conquer OS/2 !!

Now OS/2 can respond to your every whim

Introducing:



Assistant

PM Assistant is the **first** and **only** Automation and Customization Utility for OS/2 Presentation Manager

- Automate repetitive tasks with **mouse and keyboard input macros.**
- Play keystrokes to any PM application or full-screen session.
- Start, position, and initialize your PM applications **automatically.**
- Switch to and position PM applications and full-screen sessions with user-defined **hot-keys.**
- Execute any macro from within any PM application or full-screen session with user-defined **hot-keys.**

- Set, view, and change *any* program's **OS/2 process priority.**
- Assign **accelerator keys** and rearrange the menu commands of *any* PM application.
- Speed dialog box interaction with Speed Button.
- Prevent damage to your monitor with **Screen Saver.**
- Protect your computer from unauthorized access with **Password Protection.**
- Create turn-key procedures or demonstrations of your PM programs with the **demonstration utility.**

Now, all your favorite tools in Dos and Windows are available for Presentation Manager along with features unique to OS/2. PM Assistant comes with on-line help, an unconditional satisfaction guarantee, and a **free program** - DiskWatch, a disk free-space monitor. PM Assistant runs on OS/2 versions 1.10 or greater, and is now available at a special introductory price:

PM Assistant.....$125.00

Take control of OS/2, order today!

1-800-745-7757

Utilis Advanced Tools Now

Trademarks are property of their respective holders.

P.O. Box 15905
Seattle, WA 98115
(206) 467-4025

Circle 56 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Network madness

By M. Steven Baker

Network programming is a black hole for many developers. The details of networking software and hardware have been mysterious for a number of reasons. Some vendors remain tight-lipped to protect their proprietary applications. For others, documenting programming interfaces is considered a low priority. Although numerous books cover the intricacies of video hardware, hardly any cover PC network hardware in comparable detail. So I'll review some books that may help you sort out the network madness on your next LAN project. Hoist the nets and prepare to enter a sea of acronyms.

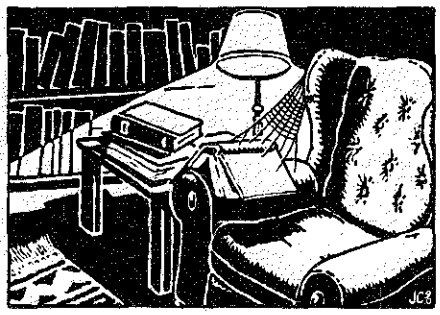


ILLUSTRATION BY JIM CARPENTER

NETWORKING UNDER DOS

Although IBM came out with its first PC network adapter card and software in 1984, documentation on NetBIOS and Server Message Block protocols wasn't really available for many years. An older book that I recommend highly is Sams' 1988 ***C Programmer's Guide to NetBIOS,*** by W. David Schwaderer. The author was a senior IBM programmer and wrote several Big Blue communication products. The book is organized into four parts, beginning with a brief history of IBM's PC LAN hardware and software and an introduction to NetBIOS. The next hundred pages cover NetBIOS support programming in detail.

Part III is an excellent treatise on cyclic redundancy check (CRC) calculations, followed by a reference section (with appendices) on NetBIOS. Bound in the back is a convenient NetBIOS reference card. The book's example programs, written for Microsoft C 5.0, are available on diskette from the author at additional cost. Schwaderer's book is often mentioned by programmers—for example, Phil Katz (of PKARC and PKZIP fame) cites the treatise on CRCs on which the PKZIP CRC implementation is based, and John Otken (crack assembly language programmer) refers to it in this issue's "Spying on NetBIOS." If you need a book that covers NetBIOS, this is still the best one to get.

Well before the arrival of DOS 3 and IBM's network products, Novell was selling NetWare. Although there are many (perhaps too many) books about managing and running NetWare, a paucity exists on programming for NetWare, NetBIOS, and DOS 3.*x* networking. Fortunately, ***Network Programming in C,*** by Barry Nance (Que, 1990), fills this gap admirably. You may be familiar with the author from his activities on BIX (*Byte*'s on-line information network). Nance's book covers network programming using DOS 3.*x* services, NetBIOS, IBM's PC LAN program, the OS/2 LAN Manager, and Novell's NetWare. The book has four major sections. The first covers basic PC networking, multiuser issues in the DOS environment, and PC-to-PC communication concepts. Four higher-level protocols are discussed: Novell's Internet Packet Exchange (IPX) and Sequenced Packet Exchange (SPX)—which were an adaptation of the Xerox Network Standard (XNS)—along with NetBIOS and Server Message Block (SMB), introduced by IBM.

Section II—the heart of the book—discusses network programming techniques. Nance details programming at the DOS level (SHARE and DOS 3.*x* features), using NetBIOS, and using IPX and SPX (NetWare) services. Useful sample code is provided to detect the presence of a LAN. These tests include checking for network disk drives, NetBIOS, NetWare, SHARE, PC LAN, and custom LANs. Code examples illustrate how to identify workstation names and user IDs and also demonstrate file sharing, locking records, and network printer services. Other chapters in Section II cover NetBIOS, IPX/SPX programming, and program testing and debugging.

Section III is a reference to DOS, PC LAN, Novell, OS/2, NetBIOS, IPX, and SPX functions. Section IV contains source code listings for several larger program examples, along with header files and network error codes. Example programs include a file and record collision tester, a NetBIOS microscope (tester), a remote program executer (and matching remote job-submission program), postman, and electronic mail programs.

Most smaller code fragments are in standard C code, and examples illustrate the differences between Microsoft, Lattice, and Turbo C compilers. The larger programming examples are based on Borland's Turbo C compiler (v2.0) using the in-line assembler. Conveniently, the sample programs are supplied on a disk bound in the back cover. If you can buy only one book on PC network programming, *Network Programming in C* is my recommendation.

Another useful new title is Prentice Hall's ***A C Programmer's Guide to the IBM Token Ring,*** written by William Roetzheim (1991). You may know Bill from his contributions to our "Consulting" column. Although any number of books cover Ethernet hardware, information on IBM's Token Ring has been scarce. This *Guide* is a welcome resource, presenting details on IBM's Token Ring hardware and software in an easy-to-understand, concise form.

Programming for the token ring is covered at varying levels of support. Chapters discuss programming at the hardware level and programming using PC LAN Redirector, NetBIOS, and Data Link Control (DLC) services. The last chapter examines transaction-oriented processing using the PC version of IBM's Application Program-to-Program Communication (APPC/PC). Code fragments in Borland's Turbo C illustrate most features and form the basis of a working library; a companion disk is available from the author. This

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

is the only book I've found that covers programming at the DLC and token-ring hardware level. For those who work with "Lord of the Token Rings," this book is recommended highly.

UNIX MEANS NETWORKING

In other environments, networking is taken for granted: Unix is synonymous with networking in many ways. Most of the first experiments with networking were based on Unix, and few Unix systems exist today without some network connection. Even Novell's NetWare protocols are based on Xerox research activities under Unix. Although good resources on DOS networking are scarce, for Unix we have an abundance of good material. I'll discuss some of the best.

Unix Networking, edited by Stephen Kochan and Patrick Woods (Hayden, 1990), is an excellent general overview of network programming in the Unix environment. The book is a collection of writings by expert authors on specific Unix network concepts. Separate chapters cover the Unix-to-Unix Copy Program (UUCP), TCP/IP, Sun NFS and RPC, AT&T STREAMS, AT&T's Transport Layer Interface (TLI) and Remote File System (RFS), OS/2 LANs, networking and the X Window System, and Sun NeWS.

Brian Redman (Bell Labs) begins the book with a useful description of UUCP. Redman was one of the authors of the HoneyDanber version of UUCP, now distributed with AT&T Unix. Douglas Comer (of Xinu fame) and Thomas Narten (Purdue) provide an overview of TCP/IP and the Berkeley socket interface. Douglas Harris (Marquette University) covers the AT&T Unix networking interfaces. These include STREAMS (which was created by Dennis Ritchie), TLI, and RFS. The OS/2 chapter, written by Martin Dunsmuir (Microsoft), discusses LAN Manager/X, including the core SMB and Extended SMB protocols and NetBIOS. Adrian Nye (O'Reilly and Associates) examines networking and the X Window System and X protocol. Louis Delzompo and Owen Densmore (Sun) describe NFS and RPC protocols and summarize the Network Extensible System (NeWS). All in all, *Unix Networking* is an expert overview of networking concepts.

Unix Networking Programming, by W. Richard Stevens of Bell Labs (Prentice Hall, 1990), is a fine textbook. The first few chapters provide a framework for the Unix process model and interprocess communication (IPC), as well as a discussion of networking and various communication protocols. Individual chapters provide details on Berkeley sockets and the AT&T System V Transport Layer Interface. The book also covers networking examples (including network library routines based on 4.3BSD Unix), security, time and date routines, Ping routines, file transfer, line printer spoolers, remote command execution, remote login, remote tape access, and remote procedure calls. A short, interesting chapter compares the performance of various Unix IPC and network mechanisms on a Compaq 386/20, a DEC MicroVax, and an AT&T Unix PC (10-MHz 68010). Although other protocols (such as XNS) are mentioned, most of the focus is on TCP/IP and the Unix network program concepts that are based on it. This practical book cites case studies from actual network applications; its extensive C source code (about 15,000 lines) further facilitates understanding.

NETWORK PROTOCOLS

For many programmers, the beauty is in the details. Although a bit dry, Sams' three-volume ***Handbook of Computer-Communications Standards,*** written by William Stallings, is good starting point for learning network protocols. Volume 1 (second edition, 1990) describes the Open System Interconnection (OSI) model and OSI-related standards. It discusses the now-familiar OSI layered model for communications and protocol details of OSI-related standards at all seven layers, including HDLC, X.25, Internet, transport, session, and presentation services. Volume 2 focuses on LAN hardware standards (IEEE 802 and ANSI), including logical link control (LLC), Ethernet (CSMA/CD), token bus, and token ring. The second edition of Volume 2 (1990) adds information on Ethernet's unshielded twisted pair (10BASET) and fiber-distributed data interfaces (FDDI). Volume 3 covers the TCP/IP protocol suite. It provides a detailed description of some of the protocol standards in this family, including Transmission Control Protocol (TCP), Internet Protocol (IP), Simple Mail Transfer Protocol (SMTP), and TELNET. Volume 3's second edition (1989) adds details on Reverse Address Resolution Protocol (RARP), Exterior Gateway Protocol (EGP), Simple Network Management Protocol (SNMP), and User Datagram Protocol (UDP). You'll find no source code here—just useful narrative descriptions and details on network protocols.

If you're interested in a discussion of programming issues, read Prentice Hall's two-volume second edition of ***Internetworking with TCP/IP,*** by Douglas Comer and David Stevens. These books are based on Comer's earlier books on the Xinu operating system, a Unix workalike. Volume I covers TCP/IP principles, protocols, and architectural concepts, and Volume II focuses on design, implementation, and internals. This volume is illustrated profusely with C code to implement the underlying TCP/IP family protocols. The earlier edition implemented ARP, UDP, and IP code. The current edition enhances this code and adds ICMP and the higher-level TCP and SNMP protocols using a socket-level interface. Be forewarned, though: the network hardware interface is for DEC Ethernet. If you need to understand TCP/IP, these books are an excellent resource.

If your networking activities extend to the Macintosh, Addison-Wesley has two helpful books. ***AppleTalk Network System Overview,*** by Apple Computer (1989), provides a readable overview of Appletalk—Apple's network system. The first part briefly describes AppleTalk hardware and software, and Part II focuses on AppleTalk protocols and applications.

The second edition (1990) of ***Inside AppleTalk,*** by Sidhu, Andrews, and Oppenheimer (of Apple Computer), gives detailed descriptions of the AppleTalk protocol architecture. This reference book covers the physical and data links, data flow, named entities, reliable data delivery, and end-user services. Appendices provide hardware specifications and software details. Like the other books in Addison-Wesley's Apple series, these books are very well produced and easy to read.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Network debugging and comm libraries

By M. Steven Baker

A simple task often unearths all sorts of interesting and time-consuming distractions. Because of this issue's theme, my plan for this space was to review some network programming software tools and to look at the newer versions of DOS communication libraries. I'll start with the networking tools, because they make a nice connection with the discussion of 386 debuggers in my May/June "Impressions."

WHEN THE GOING GETS TOUGH

Connecting similar machines is occasionally problematic, but working with divergent hardware and software from multiple vendors can create real headaches. I've been working for some time with peer-to-peer network software that allows DOS machines to interconnect with Unix and other platforms. I had intended to review several TCP/IP and socket library packages for this issue, but new releases from FTP, Sun, and NetManage arrived late, so I'll postpone this discussion.

Here at *PJ*, we use peer-to-peer network software for editorial and production. On older 386 machines we run DOS, a memory manager (to free space for network drivers), network software, and memory-intensive applications such as Ventura Publisher and Corel Draw. We occasionally experience problems that cause our network server or workstations to lock. Not being naive, I *know* that most of this software has bugs. But what is at fault—DOS, the memory manager, the network drivers, the application software, or the aging hardware? Software layering makes this even more perplexing.

At home, I network a fast 486/33 with several older machines. I have occasionally managed to lock the 486 by XCOPYing files over the network. No memory managers here, so is it a hardware or software (DOS or network) problem?

When the going gets tough, I reach for **Periscope**'s debuggers. If your work involves hardware or problematic software (device drivers and interrupt-driven functions like comm ports and networks), Periscope's hardware-assisted debuggers can't be beat. The least expensive hardware option is a simple breakout switch (to cause an NMI interrupt) and software (Model II, $225).

In some situations you may need protection for debugging code or a full 640K for programs, so option boards (Model I with 512K, $595) are available with write-protected memory. At the high end, there are pods (Model IV, from $600) that plug into your 286/386/486 socket to provide complete CPU state information—data not available without an expensive in-circuit emulator (ICE). Periscope's latest software debugger (v5.06) even has an option to work with Qualitas's 386MAX (v5.11). The EMS manager provides protected memory for Periscope, minimizing the need for a hardware board. The Periscope software is lean and powerful, with enhanced features when used on 386/486 machines. Their debugger supports Microsoft and Borland symbolic debugging information and .RTLink and PLINK overlays. For debugging device drivers and memory-resident programs, Periscope is an ideal tool.

Another option for tough debugging tasks is the handful of products from Nu-Mega Technologies. Their flagship product, **Soft-ICE** ($386), is a 386 debugger. Soft-ICE puts 386/486 DOS machines into the virtual-8086 mode, allowing control over instruction execution and memory access. Breakpoints can be set on individual port and memory accesses, allowing almost full-speed execution. Soft-ICE supports all sorts of conditional breakpoint parameters as well as I/O logging. The latest version (v2.5) understands Borland's and Microsoft's symbolic debugging information, .RTLink, and LINK overlays. CodeView can be run under their **MagicCV** product with Soft-ICE, freeing DOS memory for your program. Their latest product, **CV/1** ($129), allows debugging with Codeview on a single monitor under Windows 3.

Bounds-Checker ($249) is a valuable virtual-8086 debugging tool that catches out-of-bounds memory accesses while running DOS programs. The user creates exception files to indicate where writes to code segments, DOS, and BIOS areas are allowed. The latest version (v1.1) integrates with Soft-ICE for pop-up debugging when an out-of-bounds action occurs. Nu-Mega has developed a similar program for Novell to debug NetWare 386 Loadable Modules (NLMs).

DEBUGGING NETWORKS

If your work involves heterogeneous networks, you may eventually stumble onto a problem that requires a network protocol analyzer. Most products for analyzing networks bundle hardware (usually a portable 386 computer) with special software—a rather expensive proposition. In the process of reviewing TCP/IP implementations, I worked with FTP Software's **LANWatch** ($995), a software-only product that turns any PC into a LAN analyzer. It's available for Ethernet, StarLAN, and token ring networks—the user must supply an interface card for the network being analyzed.

Let me illustrate LANWatch's use with an example: a friend was trying to integrate several PCs (used as data-entry stations for scanning and registering livestock brands) with a dedicated Unix machine. The custom application was based on an NCR Unix box as a database server using TCP/IP. Once TCP/IP had been installed on both DOS and Unix, simple testing with FTP indicated that small files sent from a PC to Unix containing, say, "abc," ended up doubled ("abcabc"). Which network software was causing the glitch? With conventional tools, it was impossible to tell. With LANWatch, it took all of 10 minutes to log activity during the file transfer and analyze what was taking place. I've also used LANWatch to look at proprietary protocols such as Artisoft's LANtastic over Ethernet. If your problems require a protocol analyzer, I recommend LANWatch highly.

LOW-COST UNIX NETWORKING

Although DOS has many strengths, Unix wins hands down when it comes to networking. Most Unix machines can be considered part of one worldwide network. The Unix community supports a huge exchange of information, digital data, and mail transparently—using dial-up modems

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

(UUCP) and dedicated network connections (the Internet). An interesting product for DOS users is **Coherent,** by Mark Williams. This 286 version of a Unix-like clone is priced at a mere $99.

If you want to get your feet wet in the Unix environment, I recommend this product on several accounts. Coherent comes with a solid version of the Unix-to-Unix Copy Program (UUCP), yet it costs less than many comm packages. With UUCP and a friendly Unix host handy, the wealth of information available on the Unix net can now be yours. With UUCP, you queue up requests to send or receive files (mail, regular news forums, or digital data) that are modemed by your computer to a Unix host. The process is automatic and transparent, and it represents much of what Unix has to offer. Because UUCP file transfers are automated, your connect time is minimized: UUCP uses a sliding window protocol that delivers close to line speed in most situations.

Another plus for Coherent is its light overhead. Coherent runs on 286 machines with as little as 640K of memory and 8MB of free hard-disk space. Unlike many 386 Unix implementations, Coherent coexists easily with DOS. Installation takes four floppy disks and about 20 minutes compared with 20 to 80 disks (Unix V Release 4) and half a day. A Coherent disk partition doesn't have to be the first one: their custom boot sector code will default to DOS, or you can choose the partition to boot from by typing a number during the boot process. Files are easily copied to and from DOS floppy and hard disks. All this plus a C compiler and most Unix utility programs, including **lex, yacc, nroff, tar, compress,** and others.

Coherent does have its limitations: the compiler currently supports only the small model (they're working on a 386 version), and it's K&R C, not ANSI. Some hardware devices (such as network cards) are not yet supported. All in all, though, you can't go wrong for the price. One warning: like OS/2, Windows, and Unix, Coherent can consume large quantities of time because it poses a new, interesting challenge for programmers. Don't say I didn't warn you—but by all means try UUCP and Coherent as a window into the Unix world.

COMMUNICATION LIBRARIES

Because several serial communication libraries have been updated recently, I thought it would be challenging to try porting a few of these products to work with DOS extenders. These packages are available with source, so porting can be a test of clean, portable code. New features in comm libraries are in four areas: 16550 FIFO support and UART hardware flow control, file transfer protocols, terminal emulation, and network modem support. It's ironic that vendors are supporting more file transfer protocols at a time when error-correcting modems, which mitigate much of the reason for robust protocols, are becoming the norm.

TABLE 1: Comparison of communication product features

Feature	Async Manager	Essent. Comm	GreenLeaf CommLib	C Comm Toolkit	Solid Link	SilverComm Async	COMM DRV
File Transfer:							
Xmodem	X	X	X	X	X	X	N/A
Ymodem	X	X	X	X	X	X	N/A
Ymodem-G		X	X	X	X	X	N/A
Zmodem			X		X		N/A
Kermit		X	X	X	X		N/A
Sealink/Telink					X		N/A
Hardware/Interface:							
16550 UART	X	X	X	X	X	X	X
Multiport boards		X	X	X		X	X
Network APIs			X			X	X
Handshaking:							
XON/XOFF	X	X	X	X	X	X	X
Hardware	X	X	X	X	X	X	X
Terminal Emulation:							
ANSI		X		X	X	X	N/A
VT100				X	X	X	N/A
Other				X			N/A

ANSI C compilers are a wonderful improvement over K&R, but they don't fully replace a good lint program. I use a variety of C compilers for just this reason. For instance, MetaWare's High C/386 is fussy about C syntax and quiet conversions. This can be very helpful. At other times, you may prefer the raw compile speed and debugging offered by Borland's C++.

My technique for porting comm source code used **PC-Lint 386** from Gimpel Software ($239) as a starting point. This latest version has a bundled-in 386 DOS extender, so there are virtually no limits on file size or modules that can be checked. The package includes the 16-bit version of PC-Lint (for DOS and OS/2) for use under Windows 3.

PC-Lint comes with an extensive set of compiler and library dependency files that handle predefined preprocessor macros, special keywords, and library differences. The program supports an overwhelming number of options to inhibit various error options and to handle memory models, ANSI, libraries, and output. PC-Lint can also output binary lint object modules so that you need to run lint only on files that have changed. There is definitely a steep learning curve in using this product effectively. Once it's mastered, however, its benefits are hard to beat.

The communication products I worked with are listed in "Vendor Information." All but one are comm libraries that support low-level UART controls, serial I/O, file transfer, and terminal emulation. **COMM-DRV** ($89) from Willies' Computer is an interesting, inexpensive, interrupt-driven device driver for comm ports that also supports background communication. When it's used as a device driver, you can read and write to it like any other DOS device. This makes it attractive for hardware control from DOS and extended DOS programs. The ASM source can also be directly linked with your application.

Two years ago, I ported Blaise's Async Manager to 286 and 386 DOS extenders—so I'm familiar with the problems. A professional 286 DOS-extender port is more difficult than one for the 386, because dual interrupt handlers (both real and protected mode) must be used with shared buffers. Otherwise, characters might be dropped while the 286 switches between real and protected mode; the 386/486 switches quickly enough that this isn't a concern. These problems are minimized further if

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

16550 chips (with 16 byte FIFOs) are substituted for older 8250/16450 UARTs.

Blaise's **Async Manager** ($189) and Soft Mountain's **Essential Communications Library** ($329) represent the older breed of comm libraries. Both are based on an assembly language TSR that provides low-level comm services (which can also be linked in with your application). The other products are Greenleaf's **CommLib** ($359), SilverWare's **SilverComm** ($249), Magna Carta's **C Communications Toolkit** ($149), and Solid Software's **Solid Link** ($497 with source). Depending on your situation, serial I/O code might be used to handle devices (such as graphics tablets, plotters, etc.), to transfer files, or to emulate a terminal for remote communication. Your own requirements should dictate which product is right for you.

A SIX-PACK OF PACKAGES

Table 1 (p. 87) compares features of these comm packages. No single product has everything. SilverComm is strong on low-level UART control, event timers, and multiport boards. Solid Link supports the most file transfer protocols, interfaces with Modula-2, and is written in assembly language for small size and speed. CommLib offers the best support for multiport serial boards, can talk with some network modem servers, and has by far the best documentation. The C Communications Toolkit has an interface to the Z80SIO UART, supports the DCA/Intel standard (CAS) for talking with FAX boards, and offers a Televideo terminal emulator. Essential Communications is strong on multiport boards and has code to handle background file transfers. Blaise has a product similar to their Async Manager for use with Turbo Pascal.

The library source code for these products consists of a number of C files along with low-level routines in assembly language. The vendors supply source code written for ANSI C. Unfortunately, each vendor's code uses different macros and **typedef**s to handle DOS memory models for portability. Solid Link consists of 340K of ASM source in nine files with only video routines in C (60K). At the other end of the spectrum is Magna Carta's **C Communications Toolkit**, which is almost exclusively C source (310K) with minimal assembly (30K) for the interrupt service routines.

My most difficult problem was converting the complex ASM macro files; the remaining conversion of the ASM code was generally not difficult. I had the fewest ASM problems with C Comm Toolkit (small size), CommLib, and Async Manager. For example, I needed to change only about 30 lines to convert Blaise's large (120K) ASM file into a simple 286 DOS-extender library (with only a protected-mode interrupt handler). For the 386, changes were more time-consuming than difficult. All offsets had to be changed to use 32-bit registers, and the C code often required a more careful review. I had the easiest time with the C code from the two older products—Async Manager and Essential Comm—which had fewer macros. The SilverAsync C code was also easy to work with.

I found Solid Link's library design thoughtful—it used simpler names and fewer routines. It also had the most ASM code to convert—more than I had patience for. Essential Comm Lib also had too much ASM code to convert manually. The C Communications Toolkit used a large comm structure (300+ bytes), which I found awkward, but it did have the least ASM source to convert. Both Greenleaf and SilverWare have hundreds of small routines that require more work for porting but provide detailed control. I found Blaise time-consuming but easy to port. After many arduous days, I have much of Async Manager, SilverComm, and C Comm Toolkit running in 386 protected mode. The others had either ASM files that were too monolithic or troublesome C files. It's clear to me that plenty of serial I/O options are available for 386 DOS. I'll comment on the final results of my porting efforts next time. Until then, stay cool, calm, and connected. PJ

VENDOR INFORMATION

Periscope 5.0
The Periscope Company, Inc.
1197 Peachtree Street, Plaza Level
Atlanta, GA 30361
(800) 722-7006
Circle 169 on Reader Service Card

Soft-ICE 2.5 & Bounds-Checker 1.1
Nu-Mega Technologies
P.O. Box 7780
Nashua, NH 03060-7780
(603) 888-2386
Circle 170 on Reader Service Card

LANWatch v2.01
FTP Software, Inc.
26 Princess Street
Wakefield, MA 01880-9855
(617) 246-0900
Circle 171 on Reader Service Card

Coherent v3.1
Mark Williams Company
60 Revere Drive
Northbrook, IL 60062
(800) 627-5967
Circle 172 on Reader Service Card

PC Lint 386
Gimpel Software
3207 Hogarth Lane
Collegeville, PA 19426
(215) 584-4261
Circle 173 on Reader Service Card

Async Manager v3.0
Blaise Computing Inc.
819 Bancroft Way
Berkeley, CA 94710
(415) 540-5441
Circle 174 on Reader Service Card

Essential Communications Library 3.0
South Mountain Software Inc.
76 South Orange Avenue
South Orange, NJ 07079
(800) 451-6174
Circle 175 on Reader Service Card

Greenleaf CommLib 3.1
Greenleaf Software Inc.
16479 Dallas Parkway #570
Dallas, TX 75248
(800) 523-9830
Circle 176 on Reader Service Card

C Communications Toolkit v1.08
Magna Carta Software, Inc.
P.O. Box 475594
Garland, TX 75047-5594
(214) 226-6909
Circle 177 on Reader Service Card

SilverComm v3.04
SilverWare
3010 LBJ Freeway, Suite 740
Dallas, TX 75234
(241) 247-0131
Circle 178 on Reader Service Card

Solid Link v2.3
Solid Software
227 West Fourth Street
Cincinnati, OH 45202
(513) 651-1133
Circle 179 on Reader Service Card

COMM-DRV
Willies' Computer Software Co.
2470 South Dairy Ashford, Suite 188
Houston, TX 77077
(800) 966-4832
Circle 180 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

LEARN SOMETHING NEW EVERY DAY DURING SOFTWARE DEVELOPMENT WEEK '91

Boston, September 8–13
Boston Marriott Copley Place

Expanded opportunities for software developers and managers in Boston this Fall!

See the latest in development technology at the concurrent Software Development Products Exhibition!

SOFTWARE DEVELOPMENT

An all-encompassing educational experience can be yours in Boston, September 8–13, 1991. Our bi-annual software development event has expanded into a full five days of techniques and technology designed to increase your programming power. You can't afford to miss this important industry show. You'll get a year's worth of knowledge in a handful of days!

Two conferences in one!

Software Development '91—Fall

The best of Software Development '91 heads east. At Software Development '91—Fall, you'll find the same caliber of speakers that Software Development conferences are known for. Speakers like Philippe Kahn, P.J. Plauger, Bill Gates, Ed Yourdon, Bjarne Stroustrup, Steve Mellor, Brad Cox, Robert Ward, Jack Purdum, and others. We'll give you the best educational line-up anywhere on object-oriented programming, programming for Windows, C++, design, methodologies, and more!

People & Systems: The Software Development Management Conference

Chaired by Larry Constantine, this new conference for top-level software managers focuses on the increasingly important issues of quality, productivity, and management of change as it pertains to the software development field. Be sure you miss nothing. We've scheduled People & Systems for the same week as Software Development '91 so you can attend both!

PLUS in-depth one- and two-day technical focus seminars on hot topics like Windows, Windows tools, C++, object-oriented techniques, and international marketing!

YES! Send me more information on Software Development Week September 8–13, 1991 at the Boston Marriott Copley Place. I'm interested in:

☐ Software Development ☐ People & Systems

Name .....
Title .....
Company .....
Address .....
City/State/Zip .....
Phone ..... Fax .....

Software Development Conferences & Shows
P.O. Box 7843, San Francisco, CA 94120-7843
Call: (415) 905-2414 / Fax: (415) 905-2630
Send information on: ☐ attending ☐ exhibiting Code DVR

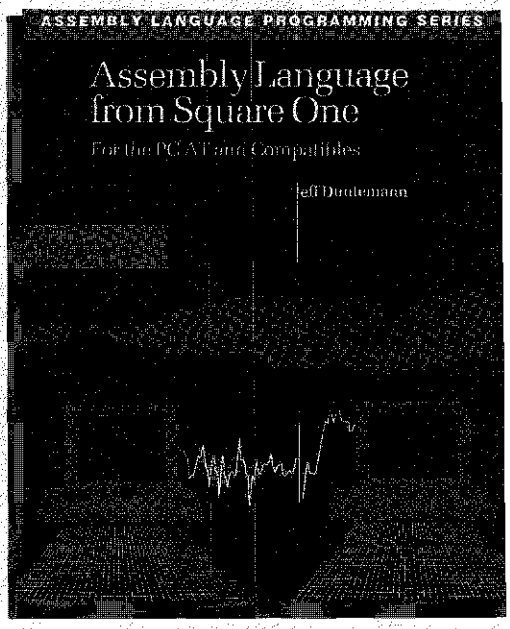
Circle 57 on Reader Service Card

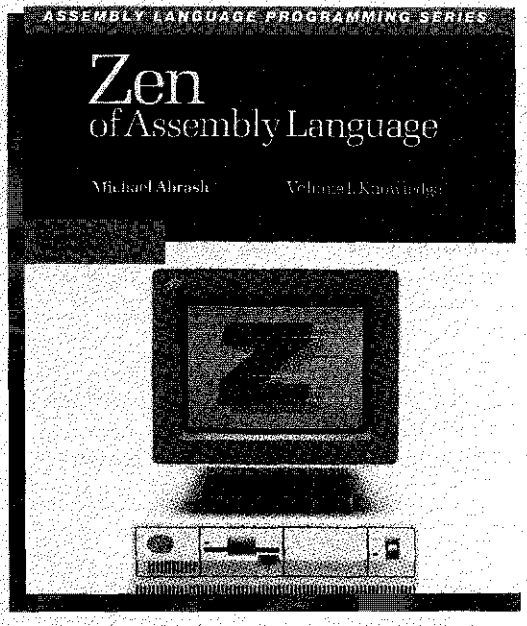
PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Best Books from PC TECH

Take It To The Limit!

Jeff Duntemann taught you object-oriented programming in the pages of *PC TECHNIQUES*, and he can teach you assembly as your first programming language. Jeff's crisp metaphors and offhand, humorous style carries even the very toughest concepts home. After reading this book you'll be writing programs in assembly, but more important, you'll know both what you're doing and why you're doing it. **Every copy signed by the author!**





Order *Assembly Language from Square One* from PC TECHNIQUES and receive the listings diskette for FREE!

"Pushing the Envelope" columnist Michael Abrash has made hand-optimized assembly code his trademark, and in this monumental volume shares the accumulated wisdom of his entire career as a performance programmer. Understand how the 80x86 family does what it does. Learn of the "cycle eaters" that slow your code down. Learn how to do what no higher-level language compiler can do. That's the Zen of assembly language. **Every serious programmer must have this book!**

Send Coupon or Call Today!—602-483-0193

Assembly Language from Square One (Free Disk) $29.95	___Check/Money Order ___VISA/Mastercard
Zen of Assembly Language $29.95	Card #_____ Exp. Date_____
Subtotal $_____	Name_____
Shipping:	Company_____
$2.50 per book in US.	Address_____
$5.00 per book outside US.	City/State/Zip_____
$_____	Country_____
Total $_____	Phone_____
Prepayment required—U.S. Banks only	PC TECHNIQUES, 7721 E Gray Rd., Suite 204, Scottsdale, AZ 85260

STOP!

Don't write another line of assembly code until you've read these books!

Circle 58 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PRESSROOM

PROGRAMMING COMPETITION

The **Johns Hopkins University** is conducting its second nationwide Search for Computing Applications to Assist Persons with Disabilities. A $10,000 Grand Prize will be awarded for the best computer program or device that assists persons with disabilities. Other prizes include personal computers and certificates of achievement. Previous winning entries have included the Apple Adaptive Firmware Card, the Reading Machine for the Blind (which converts printed materials into synthetic English speech), and Total Talk, a talking computer terminal for the blind.

The deadline for entries is **August 23, 1991.** For information, contact CAPD, P.O. Box 1200, Laurel, MD 20723, (301) 953-5364.

NETWORK C LIBRARY

The Network C Library from **Automation Software Consultants** provides 300 functions for accessing NetWare system services and statistics. The services include file and directory management, locking, synchronization, bindery management, messaging, printing, queue management, transaction tracking, and diagnostic reports and performance statistics for the file server and individual workstations. Sample programs and documentation are included. The Network C Library is $225 ($450 with source code).

Automation Software Consultants, Inc. (ASCI), 124 Venice Avenue, Cincinnati, OH 45140, (513) 677-0842.

Circle **182** on Reader Service Card

NEWT AND CHAMELEON

NetManage Inc. has released NEWT, a TCP/IP communications stack for Windows 3, along with Chameleon 2.0, a TCP/IP applications package for Windows. Chameleon is implemented as a DLL and allows users to run concurrent TCP/IP sessions with multiple hosts. Release 2.0 includes Telnet, FTP, Ping, SMTP, and other applications, along with support for Serial Line IP and IP routing. NEWT is $200; Chameleon is $400.

NetManage Inc., 10020 North DeAnza Boulevard, #101, Cupertino, CA 95014, (408) 257-6404.

Circle **183** on Reader Service Card

MASM 6.0 ARRIVES

Microsoft's Macro Assembler professional development system 6.0 provides high-level language constructs, an *n*-pass assembler, and automatic code optimizations for assembly-language programming. Features include directives that create a high-level procedure interface for defining and calling routines. Compare and flow-control directives automate the implementation of loops and decision structures. MASM 6.0 includes the Programmer's Workbench and supports Windows and OS/2 (versions 1.1 and later). It costs $150; upgrades are $75.

Microsoft Corporation, One Microsoft Way, Redmond, WA 98052-6399, (206) 882-8080.

Circle **184** on Reader Service Card

DEBUGGER FOR WINDOWS

MultiScope, Inc. has released the MultiScope Debugger for Windows. Both Microsoft Windows 3 and standard character-mode interfaces are available for debugging Windows and DOS applications written using CodeView-compatible compilers. The debugger includes runtime debugging and a postmortem capability that allows analysis after a program crash. Remote debugging allows the debugger to be run on a host computer and the application to run on a remote machine when connected via a serial line or network. MultiScope Debugger for Windows is $379.

MultiScope, Inc., 1235 Pear Avenue, #111, Mountain View, CA 94043, (415) 968-4892.

Circle **185** on Reader Service Card

VISUAL BASIC AND QUICKWIN

Visual Basic, from **Microsoft,** allows Windows interface components—such as command buttons, text fields, pictures, menus, and file system controls—to be created visually without writing code. The Visual Basic language is derived from the Microsoft QuickBasic environment. It uses a threaded p-code incremental compiler and source-level debugging tools. Visual Basic supports DDE and DLLs. The control set can be extended using C and the Windows SDK and the Visual Basic Control Development Kit, available separately. Visual Basic is $199.

Microsoft's Fortran for Windows 5.1 lets Fortran applications run in the Windows environment by recompiling and linking with the new QuickWin Fortran library. The Fortran compiler supports FORTRAN 77 and a number of IBM, VAX, and Fortran 90 extensions. Version 5.1 includes the Programmer's WorkBench and an extended CodeView debugger. Microsoft Fortran 5.1 is $450.

Microsoft Corporation, One Microsoft Way, Redmond, WA 98052-6399, (206) 882-8080.

Circle **186** on Reader Service Card

QUICK REFERENCE CARD

Cooper Software is offering a Quick Reference Card for Windows 3. The card provides a list of all 597 Windows procedures, information on which parameters to use, and easy access to the list of manifest constants. The Quick Reference Card is priced at $9.99.

Cooper Software, 3523A Haven Avenue, Menlo Park, CA 94025-1009, (415) 364-9150.

Circle **187** on Reader Service Card

CASE WITH 3-IN-1 FOR C, C++

3-in-1 C++, a CASE tool for Microsoft Windows, is available from **GUI Computer**. Designed for building object-oriented Windows screens, the 3-in-1 Prototyper includes a screen designer and C++ code generator. The screen designer provides 12 object types. A single command automatically compiles and links the screen. C++ source code and resource, header, and make files are generated automatically. 3-in-1 is priced at $159 (C version), $199 (C++ version), or $259 for both versions.

GUI Computer Inc., 6604 Fernshaw Drive, Dallas, TX 75248, (800) 900-8010.

Circle **188** on Reader Service Card

GFA-BASIC FOR WINDOWS 3

GFA Software Technologies has introduced GFA-BASIC for Windows 3. GFA-BASIC provides hundreds of BASIC commands, with more than 400 additional commands and functions designed for Windows operations. Users can work with multiple-document interfaces, bitmap file loading, clipboards, DDE, and DLLs. GFA-BASIC allows the use of

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

EMS and can run in real or protected mode. C and assembly language routines can be bound into GFA-BASIC programs. GFA-BASIC costs $449 (286 version) or $495 (386 version).

GFA Software Technologies, Inc., 27 Congress Street, Salem, MA 01970, (508) 744-0201.

Circle 189 on Reader Service Card

SCREEN DESIGN KIT

The Screen Design Kit from **Butler Computer Systems** can be popped up inside applications to show the coordinates and the attribute byte of any position on the screen. A color palette displays available color combinations for color and monochrome monitors. The Screen Design Kit works with any programming language on IBM PCs and compatibles. It costs $99.95.

Butler Computer Systems, P.O. Box 5306, Walnut Creek, CA 94596, (415) 256-8401.

Circle 190 on Reader Service Card

DATAWINDOWS C LIBRARY

Greenleaf's DataWindows C library, a user interface toolkit, now supports extended DOS 386. Enhanced video routines in C can either access video memory directly or write to the display through calls to BIOS functions. The DataWindows library supports DOS, Unix, Xenix, and OS/2, as well as extended DOS/386. The price is $799.

Greenleaf Software, Inc., 16479 Dallas Parkway, Suite 570, Dallas, TX 75248, (214) 248-2561.

Circle 191 on Reader Service Card

CODEBASE++ FOR dBASE

CodeBase++ 1.01 from **Sequiter Software** is a multiuser dBASE-compatible class library for C++ programmers. CodeBase++ comes with a DLL for Microsoft Windows 3, allowing development of applications that are compatible with the data, index, and memo files of dBASE III through IV. It can be used with C++ compilers under DOS or OS/2. CodeBase++ costs $295.

Sequiter Software Inc., Suite 209, 9644 - 54 Avenue, Edmonton, Alberta, Canada T6E 5V1, (403) 448-0313.

Circle 192 on Reader Service Card

B-TREE FILER 5.2

B-Tree Filer 5.2 from **TurboPower Software** is an upgrade of the company's library of database routines for Turbo Pascal. Single-user and network procedures have been merged into one, reducing code size and opportunities for programming errors. EMS memory can be used for index page buffers, freeing up normal DOS memory for other uses. A utility is included that performs many of the changes to convert old files. B-Tree Filer 5.2 is $139 (single-user) or $189 (network). Upgrades are available.

TurboPower Software, P.O. Box 66747, Scotts Valley, CA 95067, (408) 438-8608.

Circle 193 on Reader Service Card

STORK INSTALLATION SOFTWARE

Island Systems has introduced The Stork installation software. All installation procedures are done using CASE tools, so no programming or script files are required. Because the Stork modularizes the installation definition process, adding product revisions is easy. The Stork uses standard IBM text mode and costs $175.

Island Systems, 7 Mountain Road, Burlington, MA 01803, (617) 273-0421.

Circle 194 on Reader Service Card

LINNAEUS

Zircel Software offers Linnaeus 1.0, a DOS C++ class browser and application development tool. It builds an information database from any subset of C++ source on multiple disks or directories. Classes or lists of classes can be selected based on regular expressions or on class, parent, or child characteristics. Graphical trees of ancestor and descendant hierarchies are possible, as is detailed class information in windows. Linnaeus works with any C++ source. It costs $195.

Zircel Software, 2285 #A South Jasper Way, Aurora, CO 80013, (303) 750-9543.

Circle 195 on Reader Service Card

c_pslib C DEVELOPMENT LIBRARY

Barton Creek Software has released c_pslib, a C development library for creating PostScript output. The library implements PostScript calls at three levels: PostScript procedures can be called directly; a midlevel interface handles graphics, text, and loops; and a high-level interface generates full PostScript output. c_pslib costs $195.

Barton Creek Software, 2222 Western Trails, Suite 106, Austin, TX 78745, (512) 441-8354.

Circle 196 on Reader Service Card

BRIEF 3.1

Solutions Systems is shipping version 3.1 of the BRIEF text editor. New features include mouse support, EMS caching, redo, enhanced window capabilities, dialogue-box support, and support for Microsoft C 6.0 Advisor. BRIEF 3.1 provides a pop-up menu and submenus that can be set to execute commands from within the window on display. BRIEF supports 42 compilers and can be configured to support others. BRIEF 3.1 is $249.

Solutions Systems, 372 Washington Street, Wellesley, MA 02181, (617) 431-7445.

Circle 197 on Reader Service Card

BBROWSE C BROWSER

The Software Annex has introduced BBROWSE, a C source-code browser that can be accessed from within BRIEF. BBROWSE presents symbol information from all files in a project. Available reports include definitions, references, function and inverted calltrees, unused symbols, and symbol details. The reports serve as home base for hypertext editing. BBROWSE is priced at $89.

The Software Annex, 3143 West 12th Avenue Court, Broomfield, CO 80020, (303) 465-5277.

Circle 198 on Reader Service Card

FORTRAN-TO-C TRANSLATOR

FORTRAN 77 Programmer's Assistant II from **Microtools** includes a Fortran-to-C translator, a runtime library, and tools to support the conversion and maintenance of C code. The translator generates K&R or ANSI C code, and the library provides more than 200 functions. The Programmer's Assistant II costs $475. The Plus version adds an option to produce structured FORTRAN-8*x* output; it costs $625.

Microtools, Box 2745, Santa Clara, CA 95055-2745, (408) 296-3478.

Circle 199 on Reader Service Carrd

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PJ BILLBOARD

FIGHT PIRACY & PROTECT YOUR PROGRAM $$$'s!

Since 1986, companies worldwide have been choosing Az-Tech security products. If you demand the strongest protection available, why not choose one of these "proven leaders".

- **EVERLOCK** Copy Protection
- **EVERTRAK** Software Security
- **EVERKEY** Hardware "Key" Software Security

For IBM and Compatibles.
30 day money back guarantee.
Free info and demo disk available.

Az-Tech Software, Inc.
305 East Franklin
Richmond, MO 64085
(800) 227-0644 (816) 776-2700
FAX (816) 776-8398

Circle 110 on Reader Service Card

QuickBASIC 4.5 - PDS 7
Software Development System

- Multiple Overlapping Windows with Full Mouse Support.
- Windows can be clicked, moved, resized, scrolled, etc.
- Popup & Pulldown Menus, Lists, Calculator, Calendar, Directories, and more.
- Dialog Boxes and Data Entry Screens.
- Context Sensitive Help Screens.
- Event-Driven Architecture.
- Supports MDA-VGA text modes.
- Integrated Development Environment.
- File, Directory, Keyboard, BIOS, and TSR functions.
- Disk-based Documentation.
- Only $59.95 (+CA Tax & $5.00 S/H)

DSE Software Publishing
Post Office Box 96 - Willits, CA 95490
707 459-4358

Circle 111 on Reader Service Card

Duplicating disks? Read this:
If not, please forward to someone who is.

EZ-DISKCOPY PRO
Duplicates Disks Fast!

- Read source disk only once.
- Quickly & flawlessly duplicates 360K, 720K, 1.2M, 1.44M disks on your own PC/XT/AT/etc. ©
- Up to 300 disks/hr/machine!
- Formats, copies, verifies, stores images (optionally "crushed"), etc.
- Pre-formatted are even faster.
- More Speed? Just add machines.
- Serialize and print labels too!
- Replaces $1000s in hardware.

$139+S&I (CP•1; $69ea. add) $495 NCP•SL

EZX P.O. Box 58177, Webster, TX 77598

Discover, VISA, MC, AX Accepted
713.280.9900 FAX:280.0525

TestDrive with Full Manual is only $10 +S&I

Circle 112 on Reader Service Card

New Version 2 - Execute up to 256 C Functions Concurrently in One Program with Conquerrent C

- **Great for real-time, data acquisition, clock displays, background printing, simulation, process control, games.**
- **Transparent reentrancy protection for most large model library functions.**
- **Flexible stack location and size management for each task.**
- **All memory models supported in Turbo and Microsoft C on DOS 80x86 PC's.**
- **New Low Price - $45 complete, no royalties, 30 day satisfaction guarantee.**

Available from:
Interstitial Software Systems
321 E. Buckingham Rd., Ste. 140
Garland, Texas 75040

Circle 113 on Reader Service Card

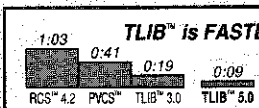
Quick and Clean' COBOL

The COBOL Report Activator *(COBRA)*:

- Generates efficient, structured COBOL programs
- Is easy to learn and use
- Accepts your COBOL code and copybook members
- Produces audit trail reports
- Uses one or two file input
- Allows site defined defaults
- Available in mainframe & PC

Olivadoti Associates
(Formerly *COBRA* Systems)
P.O. Box 750817
Petaluma, CA 94975-0817
(707)763-5123

Contact this advertiser directly



TLIB™ is FASTEST!

Times are to update a 45K library on a PC/XT. PVCS and TLIB 3.0 are from Sept 87 *PC Tech Journal*. MKS RCS 4.2 and TLIB 5.0 are newer

TLIB™ is *BEST!*

"Do not be fooled by the fact that this is the least expensive of the five packages reviewed here - TLIB has features and power to spare" John Rex, **Computer Language**

"TLIB is a great system" J. Vallino, **PC Tech J**

- **Full-Featured Version Control for Software Professionals.** Check-in/out locking. Branching. Keywords. Wildcard and list-of-file support. Can merge parallel changes and undo intermediate revisions. Network and WORM support. **Mainframe compatible** deltas for Pansophic, ADR, IBM, etc.. Integrates with **Opus™ MAKE** & **Slick™ MAKE**.

MS-DOS $139, OS/2 $195 + shipping Visa/MC
5 station LAN license $419 (OS/2 $595), call for other sizes

BURTON SYSTEMS SOFTWARE
PO Box 4156, Cary, NC 27519 (919) 233-8128

Circle 115 on Reader Service Card

Why you want BATCOM!

BATCOM is a batch file compiler that compiles your ".bat" files to ".exe" files to make them faster, more professional, and more capable. **BATCOM** extends DOS with new commands so you can read keyboard input, perform arithmetic, use subroutines, and much more. In addition, **BATCOM** protects your source code, and you can distribute your compiled programs without royalties. Only **$59.95**

ORDER TODAY!

Wenham Software Co.
5 Burley St.
Wenham, Ma. 01984
(508) 774-7036

Circle 116 on Reader Service Card

BASIC AND C PROGRAMMERS

INDEX MANAGER ®
B+ Tree File Indexing
Simple, powerful, affordable

New QuickC/TurboC version $99
QuickBASIC version $59
BASIC PDS 7 version $99
Network version $250

CDP Consultants
1700 Circo Del Cielo Drive
El Cajon, CA. 92020
(619) 440-6482

Contact this advertiser directly

FORTRAN

FORTRAN Toolkits
Comprehensive libraries of over 300 graphics, text, and systems service subroutines from ellipses to child-processes to I/O verification. High-speed video drivers, from CGA to Super VGA, are all internal. Option rich, easy to use. No royalties. 300+ pg. manual, 8,000 lines of sample source. Price $215.

DOCUMENT'er and DIAGRAM'er
Makes source code understandable. Indexes, cross-references, and diagrams the structure of source code. No changes to source required. Nested includes, recursion, and annotations handled automatically. Price $165.

IMPULSE *Engineering*
PO Box 190206,
San Francisco, CA 94119-0206
(415) 788-4611

Circle 117 on Reader Service Card

THE CD-ROM DEVELOPER'S LAB™

Comprehensive CD-ROM multimedia production resource for PC and Apple developers. Proven design, management, data prep, programming, premastering and manufacturing techniques. Important specs from leading companies. Demos of off-the-shelf tools for imaging, audio and animation. Sample applications demonstrate the power of our source tools in C, Turbo Pascal. PC or MAC.@ $395.00; Transportable version @ $425.00, Visa/MC accepted.

SOFTWARE MART, INC.
3933 Spicewood Springs Rd., Suite E-100
Austin, Texas 78759
(512) 346-7887 FAX: (512) 346-1393

Circle 118 on Reader Service Card

Meijin++
C++ Modeling & Simulation Class Library

Enter the world of Modeling and Simulation the fastest & easiest way... with class!

- Reusable Data Structures
- Statistical & Math Classes
- Exceptions
- Discrete Event Simulation
- *C++ 2.0 Documented Source*
- No Royalties - User's Manual
- Introductory Price: $225.00

Network Integrated Services, Inc.
221 West Dyer Road Santa Ana CA 92707-3426
Tel 714-755-0995 Fax 714-433-2347

Circle 119 on Reader Service Card

INTEL® 80486 MACRO DISASSEMBLER

MD86 (Masterful Disassembler) is the most comprehensive **macro** disassembler on the market. Interactive by design not after-thought. Features include:

- ▶ User defined macros with arguments
- ▶ Auto code/data separation
- ▶ Auto commenting and label naming
- ▶ Command menus and help screens
- ▶ Customizable disassembly options
- ▶ Comprehensive, indexed, manuals
- ▶ Fully supports 8086/87 thru 80486/87
- ▶ COM/EXE/SYS/*memory/other*
- ▶ Cross reference and more!

Still only $67.50 + tax ($1.50 s/h)

C.C.SOFTWARE
1907 ALVARADO AVE.
WALNUT CREEK, CA 94596
(415)939-8153

Contact this advertiser directly

Get the attention your product deserves.

Placing an ad in **PJ BILLBOARD** is an easy, low-cost way to reach potential buyers. Display ads are 1⅝" wide by 2⅜" high. For more information or to reserve a space, call our ad department at **(800) 234-0386.**

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PJ BILLBOARD

PJ CODE IS AVAILABLE!

All the source code from the articles in this issue is available on disk from

Programmer's Journal
P.O. Box 70167
Eugene, OR 97401-0110
800-234-0386
503-747-0800

Code disks are $ 8.00 each. Please see page 81 for ordering information.

PJ Code Diskscriptions are available for $39.95. Look on the card opposite page 16 to order.

The code listings are also available for downloading from several Bulletin Board Systems around the country. See below for listings.

Contact this advertiser directly

FORTRAN PROGRAMMING TOOLS

FORWARN -- lint for Fortran, plus calling trees and cross references. **DEVT TOOLS** -- pretty printer, GOTO restructurer, etc.

For DOS, $399. and $149.

Also for OS/2 and Unix.

Quibus Enterprises, Inc.
3340 Marble Terrace
Colorado Springs, CO 80906
(719) 527-1384

Circle 120 on Reader Service Card

_miniEd — Add Text Editing to Your 'C' Applications

Ultra-small (less then 5K), full featured window text editor designed to work with 'C' programs. _miniEd can be linked with your 'C' programs, or can be configured as a TSR and accessed via a software interrupt. Ideal for obtaining text input when RAM considerations are important. All text and formatting information is contained in a 'C' structure that is shared between your program and _miniEd.

+Written in 8086 assembler
+Uses any program defined window size
+No Royalties
+Non-destructive VIEW mode
+Re-Entrant/Edit multiple windows

Other language interfaces are available.

_miniEd Object, Commented Source and Documentation $99.00

Phone: 708-778-6060
Fax: 708-778-6063
Strategic Software Designs, Inc.
6 S 235 Steeple Run Drive – Suite 12B
Naperville, IL 60540

Circle 121 on Reader Service Card

If Your App Never Swaps, Stop Reading This Ad

However, if you are beset by other apps stealing *your* memory, or by real mode users, then your app occasionally crawls because of disk thrashing. There simply isn't enough memory to run comfortably, even with virtual memory and object caching.

If this sounds familiar then you need ***SegMentor***™. It optimizes your segmentation map so your app screams even when your user's memory is scarce.

SegMentor™ traces the entry point, exit point, and timing data for absolutely every one of your functions in a typical runtime session[s.] * Then, it employs powerful algorithms, like those for auto-routing printed circuit boards, to explore the solution space for an optimal segmentation map. You get a more compact app with the fewest inter-segment [disk thrashing] calls. We even put the pragma directives for this new map in an include file for easy linking back into your code.

[illegible]

*If this profiling looks illuminating compared to your statistical analyzers and source code filters, call for more information on ***Performance Tracer***™. It gives you a **precise** and **complete** view of **exactly** what occurs in your app at runtime — call trees, object sizes, call counts, function times, etc.

MicroQuill INC. ***Performance Tuning Tools***
4900 25th Ave. NE 206 • Seattle, WA 98105
800-441-7622 • 206-525-8218 FAX 206-525-8309

Circle 122 on Reader Service Card

DEBUG WITHOUT SOURCE

Quaid Analyzer debugs by showing interrupts as they occur. You can even watch DOS boot load without hardware changes. The manual tells how to deal with programs that don't want to be traced - we use it on copy protection. You scroll through memory like text in an editor, and never type a command. US $200.00

Quaid Software Ltd Dept PJ
45 Charles St East Third Floor
Toronto Ontario Canada M4Y 1S2
(416) 961-8243
FAX: (416) 961-6448

Circle 123 on Reader Service Card

LOOK

at the savings in development time you can get with DOCZ to automate subroutine and program documentation. Discover how to use DOCZ to build an object-oriented development environment to share and re-use your software modules. **FREE** MSDOS demo disk and samples. 60-day money-back guarantee. VISA/MC accepted. $195/Single-User. Start benefiting NOW!

DOCZ™
Do It Better With Toolz!
Software Toolz, Inc.
8030 Pooles Mill Drive
Ball Ground, GA 30107-9610
(404) 889-8264
Voice-mail: (800) 889-3878

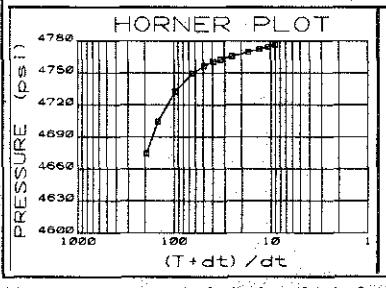
Circle 124 on Reader Service Card

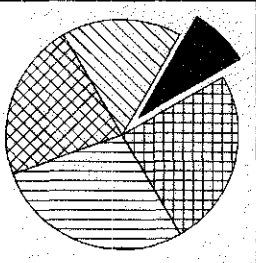
INGRAF 5.00

A Multi-Device Graphics Library for Scientific, Engineering and Business applications. Supports Video(CGA, EGA, VGA & Hercules), Printers (EPSON, Okidata, HP LaserJet, HP Paintjet & Postscript) and Pen Plotters(HPGL and DM/PL).

C, FORTRAN-77, PASCAL, MS Professional BASIC and QuickBASIC bindings.

FULL SOURCE CODE • NO ROYALTIES • $350.00





Sutrasoft

10506 Permian Dr.
P.O. Box 1733
Sugar Land
TX 77487-1733
U.S.A.
Tel: 713-491-2088
Fax: 713-240-6883

Circle 125 on Reader Service Card

The $20 C/C++ Object-Based eXecutive

Try the most comprehensive and cost-effective multitasking executive library available!

OBX can coexist with DOS and allows your PC to execute tasks simultaneously at a simplicity you just wouldn't believe.

OBX is a low cost executive library for software development in an industrial low speed real-time environment.

OBX is also a valuable educational tool for programmers tha want to learn the concepts of real-time programming.

OBX/C	$20	**Cnapse**
OBX/C w/Source	$90	*The Real-Time Contact*
OBX/C++	$30	11 Laurier, Chambly
OBX/C++w/Source	$140	Qc, Canada J3L 9Z7
		(514) 447-7221
		Fax: (514) 447-7297

No royalties, no charge for AIR shipping, and 30-day money back guarantee. All orders must be prepaid in US funds. Send a check or money order payable to Cnapse.

Circle 133 on Reader Service Card

FORTRAN TO STRUCTURED C / FORTRAN TRANSLATOR

FORTRAN PROGRAMMER'S ASSISTANT II converts spaghetti FORTRAN into neat, maintainable C code. Run-time library source and C tool set included. It will make your conversion process easy and will significantly reduce your conversion and future maintenance costs. ASSISTANT II Plus also produces structured FORTRAN. Easy to use. At low prices. A must re-engineering tool for FORTRAN programmers.

MICROTOOLS
P.O. Box 2745 PJ
Santa Clara, CA 95055-2745
(408) 296-3478
FAX (408) 243-7688

Circle 134 on Reader Service Card

Imagine what 16 million new buyers could mean to your software business...

Cash! When you put your new product in the Selective Software Catalog your sales will jump because you get:

- **Buyers:** you get buyers not lookers and you keep the names of your future good customers.
- **Circulation:** we mail 3-4 million catalogs every 2 months.
- **Free ad production:** get pro quality ads at a fraction of the cost of an agency!
- **Service and support:** we handle all sales, top-rate tech. support and fulfillment.
- **Disk dupe & packaging:** we can help you with expert package design & disk duping.
- **Payment systems:** co-op plans for first time developers.

Selective *software*
Call: 408-423-3556 Ext. 200 for more details

Circle 135 on Reader Service Card

ALL OUR CODE — ON DISK

Avoid listing errors and the on-line time of bulletin board services. **PJ Diskscriptions** are a great way to get valuable source code — and we'll deliver it right to your door. Only $39.95 for a one-year subscription. Look on the card opposite page 16 to order.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PJ BILLBOARD

OPT-TECH SORT/MERGE

Extremely fast Sort / Merge / Select utility. Run as an MS-DOS command or CALL as a subroutine.
Supports most languages and filetypes including Btrieve and dBase. Unlimited filesizes, multiple keys and much more!

MS-DOS **$149.**
OS/2, UNIX **$249.**

Opt-Tech Data Processing
P. O. Box 678
Zephyr Cove, NV 89448
(702) 588-3737

Circle **126** on Reader Service Card

QUALITY TOOLS

Memory Overlay Manager - transform your C, Clipper or Clarion application into a 5K TSR with hot-keys.....$99.95
Memory Master - break 640K barrier, one heap or many, EMS/XMS.....$59.95
ASMplus - structured ASM lang. & lib. Code ASM just like C or Pascal....$99.95
"C"erious Toolkit Plus - over 250 highly optimized fcts, C & ASM source...$99.95
386PM protected mode ver.....$199.95
Phase One - 100% replaceable plug-&-play C engine for Paradox, runs 4-10X faster, DOS and DDL ver. incl.....$495.00
PALCom - PAL lang. compiler. Create stand alone .EXE programs.....$1495.00

Please call our BBS at 516-331-6682 to try out our FREE demo programs.

TSR Systems Limited
116 Oakland Avenue
Port Jefferson, NY 11777
Ph: 516-331-6336 FAX: 516-331-6377

Circle **127** on Reader Service Card

BGI Printer Drivers

The BGI Printer Driver Toolkit provides BGI drivers for Epson 9 & 24 pin, HP LaserJet II, and HP PaintJet printers. Use the BGI graphics interface to effortlessly generate high resolution hardcopy with Turbo C, TC++, and Turbo Pascal. **Full driver source code included. $89.95**

PC Timer Tools

60 functions implement microsecond resolution timing, precision delays, interrupt profiling, and timer tick management. Ideal for profiling and data acquisition/process control. Supports TC, TC++, MSC, TP. **Full library source code included. $49.95**

Prices postpaid USA, elsewhere add $4
VISA & MasterCard accepted.
30 day money back guarantee

RYLE DESIGN
Purveyors of Big Science
PO Box 22, Mt. Pleasant MI 48804
(517) 773-0587 CIS:73047,1765

Circle **128** on Reader Service Card

The $25 Network

Try the 1st truly low cost LAN

- Connect 2 or 3 PCs, XTs, ATs, PS/2s
- Uses serial ports and 5 wire cable
- Runs at 115K baud, up to 90 feet
- Transfer 8500 bytes per second (ATs)
- Runs in background, totally transparent
- Share any device, any file, any time
- Needs only 14K of ram
- Replace all file transfer software
- $25 per network, NOT PER NODE!
- OVER 15,000 SOLD WORLD WIDE

Skeptical? We make believers!

Information Modes
Drawer F, Denton, TX 76202
817-387-3339 Technical
1-800-628-7992 Orders

Circle **129** on Reader Service Card

Windows Tools

& other huge collections of all the PD/Shareware files you need for software development, ZIPed & indexed for busy PC pros. Computer Language called our dBase product *"possibly the most productive $100 you'll ever spend"*. **30 day money back guarantee!**

Library	Disks/Files	Price
Windows	35+/300+	$99.50
XBase Lang.	87+/1550+	$149.50
C	58+/450+	$99.50
C++ (subset)	10+/59+	$49.50
Turbo Pascal	38+/380+	$79.50
Assembler	20+/250+	$59.50

We have similar products for WordPerfect, 1-2-3, Games, AutoCAD, DTP, Netware, DOS, and 22000 record PC product database.

EMS Professional Shareware
4505 Buckhurst Ct.
Olney, MD 20832 USA
(301) 924-3594 Fax: (301) 963-2708

Circle **130** on Reader Service Card

C/ANALYST

Can quickly bring you up to speed on unfamiliar C source code.

Call (503) 581-5622
For Free Demo

CATER SOFT WARE

2182 Westfarthing Way NW
Salem, OR 97304

Contact this advertiser directly

PL / 1 on DOS?

- ANSI Standard
- Native Mode Compiler
  Off load your mainframe development to PC's

DISCUS Distribution
408-655-4505

Circle **131** on Reader Service Card

UPPER DECK EDITOR
FOR WINDOWS 3.0

Regular expressions, search across multiple files, full multi-level UNDO, Windows clipboard support, keyboard macros, compile within editor, customizable. Introductory price $75 plus $5 S/H (outside USA $15). CA residents please add sales tax. 30-day money-back guarantee.

Upper Deck Systems
P.O. Box 2751
Escondido, CA 92033
(619) 741-1075

Circle **132** on Reader Service Card

PJ Code is available on the following bulletin board services:

Buffer Board
538 Cardinal Way
Springfield, OR 97477
(Data) 503-747-7636
The Buffer Board contains all of PJ's code from Sept/Oct 1987 issue through the current issue.

PCS-49 on CompuServe
Section 10
Sponsored by:
MISOSYS, Inc.
P.O. Box 239
Sterling, VA 22170
(Voice) 703-450-4181

The Silicon Zone
4419 W. Hillsboro Blvd.
Suite 290
Coconut Creek, FL 33073
(Data) 305-360-9376

Generation Five Inc.
12061 Tech Road
Silver Spring, MD 20904-9998
(Voice) 301-680-3600
(Data) 301-680-9300
Accessible via PC Pursuit, Washington, D.C.
From Files submenu, select "P" for Personal Downloads.

Autumn Hill Software Inc.
1145 Ithaca
Boulder, CO 80303
(Voice) 303-494-8865
(Data) 303-494-6868

The Programmer's Corner
5430F Lynx Ln. #224
Columbia, MD 21044
(Data) 301-596-1180, 621-4636, or 995-3744

A.C.M. Phoenix Chapter
6408 East Palm Lane
Scottsdale, AZ 85257-2563
(Data) 602-970-0474
(Voice) 602-990-3102

Gilmore Systems of Southern California
5447 Indian Hills Drive
Simi Valley, CA 93063
(Voice) 805-582-1360
(Data) 805-582-9306

Mnematics Videotex
P.O. Box 19
Sparkill, NY 10976-0019
(Voice) 800-322-3633
(Voice) 914-365-0184 in N.Y.
Mention *PJ* when you call.

Coming Next Issue: OPERATING SYSTEMS

September/October 1991 Vol. 9.5
Advertising space reservations are due:
July 5, 1991
Materials are due:
July 12, 1991

Please call 503-747-0800

Ask for Jay in the advertising department to reserve your space.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Benchmark SPECulations

By Hal W. Hardenbergh

Two years ago, a group of Unix computer systems manufacturers got together and agreed to a standardized benchmark. This group was led by the folks building systems based on the classic RISC CPUs—the SPARC, MIPS, and 88000 chips. Each expected to have the best performance.

Their consortium, known as the Systems Performance Evaluation Cooperative (SPEC), wanted to provide a single-number measure of the performance of commercial Unix-based computer systems. This single number, called a SPECmark, is the geometric mean of 10 real-world, public domain application programs that run in compiled C. Four of these programs are considered integer-only, and the other six are floating-point-intensive to varying degrees. All exercise the CPU, cache, and DRAM memory. None exercises the disk, screen, or other forms of I/O. The SPECmark number is normalized to the VAX 11/780; a SPECmark of 12 means that the system is 12 times faster than the 11/780.

The quarterly SPEC newsletter publishes the results of the 10 benchmarks, as well as their harmonic mean. Although such an industry consortium is legal under U.S. antitrust laws, it's not legal for such consortia to provide prices. So the SPECmarks of Control Data CYBER mainframes, Stardent superminis, and PC clones are published side by side. Unless one knows all the prices, one cannot infer price–performance data.

Interestingly, the manufacturer has to provide SPEC—on a standard Unix tape format—a copy of the public domain programs that constitute the suite. The compiler, revision level, and system tuning parameters must be specified. And *anybody* can purchase a copy of the tape! *Cheating is encouraged!* Any increase in performance will benefit the end user, and that's what benchmarks are for, right? So there you have it, folks: a perfect benchmark for CPU performance running compiled C under the Unix operating system.

The second SPEC newsletter (*Winter '90*) revealed that the best (and most expensive) SPARC, MIPS, and 88000 systems scored 17.6, 17.6, and 17.8, respectively. So much for visions of industry superiority. That was 18 months ago. Sigh. The thing about benchmarks is that people learn how to cheat on them early on, and they cheat often. The first cheat was to bundle up 100 poorly performing systems in close proximity and to tie a piece of kite-string around the group. This created a multiprocessor system with a "throughput" that is 100 times greater, but is no faster for any one user. This heresy was quickly shunted aside with a "throughput" SPECmark, which is ignored by everybody except systems administrators.

ILLUSTRATION BY JIM CARPENTER

The next two heresies occurred more or less concurrently. Observing that all current-generation systems have about the same integer performance, Intel publicized the performance of the i486 based on the integer-intensive subset of the SPEC benchmark suite. This performance (18.2 MIPS for a 486/33) is very close to that of the more-expensive RISC boxes and considerably superior to that of the less-costly RISC boxes that people actually buy. Sun's IPC racks up 9.6, the SPARCstation 1+, 12.4, and the DECstation 3100, 11.8. The RISC vendors, led by MIPS, screamed bloody murder.

While the RISC folks were frantically trying to shore up their integer flank, IBM nuked their floating-point flank. The bomb was the RS6000 model 320—$12,900 with disk—with an overall SPECmark of 24.6. It got that mark by turning in an outstanding performance of 32.4 in the fp-intensive portion of the suite, while providing integer performance lower (!) than that of the 486/33. And the RS6000's price was a *lot* lower than those of the faster systems offered by other RISC vendors.

The traditional RISC vendors, led by MIPS, wailed that IBM unfairly emphasized fp performance. Actually, IBM did no such thing; the RS6000 performance was reported in full conformance with the rules and procedures set up previously by SPEC and approved by all members, including MIPS. The RS6000's secret is its ability to do an fp multiply-accumulate in a single clock cycle, an ability shared by three different DSP chips and by the i860 (if you assume the i860 is not a DSP chip).

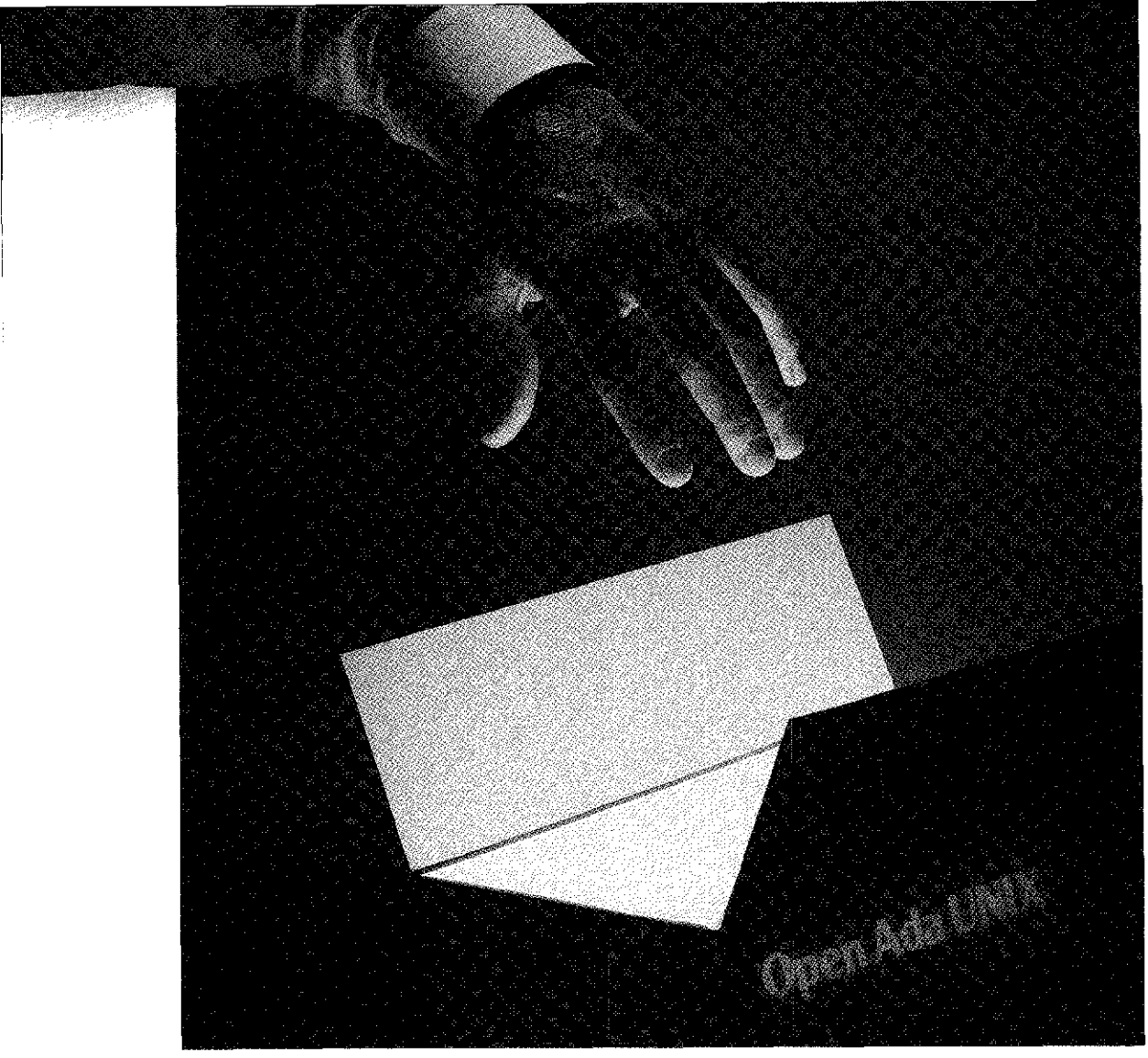
While the traditional RISC vendors frantically looked for a way to disqualify the 486 and RS6000, HP was preparing to slam-dunk the entire field with their diskless $12,000 "snake" model. The HP system has an implementation that is flatteringly similar to the RS6000's, but a SPECmark of 42.5! To improve performance, HP used a faster clock and an integer unit that effectively executes two instructions per clock by using very complex integer instructions. That means "snake" has about equal integer and fp performance.

A vast silence now emanates from the traditional RISC camp. They're beginning to regret placing the RISC label on the RS6000 and "snake." These two units have superior performance because they are more complex than the simplistic RISC chips and hence get more work done per clock cycle! The SPARC, MIPS, and 88000 are demonstrably inferior computing engines. We have the SPEC benchmarks, officially approved by the RISC folk, to prove that point conclusively.

I wonder what the traditional RISC camp will do about Intel's 100-MHz 486 working prototype, with its potential integer SPECmark of 55 or so? Furthermore, does the thought of Intel's upcoming 586 give the RISC folks nightmares?

Benchmarks are ever so much fun. PJ

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



OpenAda UNIX

Reach into the future of Ada/UNIX development.

$495 SPECIAL OFFER

Invest in an Ada/UNIX development environment now, or you'll lock yourself out of a rapidly expanding market. **As of June 1, 1991, Ada is the required language for all Department of Defense software.*** Combined with the federal government's widespread acceptance of UNIX as the operating system of choice, this requirement spells opportunity to all developers of Ada/UNIX software.

You can unlock these opportunities with OpenAda UNIX for just $495.

Professional—Despite its unprecedented low price, OpenAda is a serious development environment. In fact, professional programmers took the DOS version very seriously, snapping up over 10,000 copies within the product's first six months.

Proven—And this PC 386 UNIX version is a fully validated (ACVC 1.11) and proven environment. It's an expanded version of Meridian Software System's higher priced UNIX 386—a system utilized in such projects as FAA/OATS.

No other PC/UNIX system can touch OpenAda UNIX's capabilities without a drastic increase in price. This product is an incredible value.

Complete—Compatible with AT&T® System V version 3.2, OpenAda UNIX operates with Interactive Systems®, MicroPort®, and the SCO® version of PC UNIX. It includes:

■ Ada development environment ■ compiler/linker ■ optimizer ■ utility libraries ■ debugger ■ syntax directed editor ■ MUSIX (POSIX UNIX binding) ■ supports optional Xlib** ■ on-line LRM

Limited Offer—To remain at the forefront of software engineering you've got to explore this market and you've got to act now. **This special offer lasts just 60 days.** After that, it'll cost you $1995.

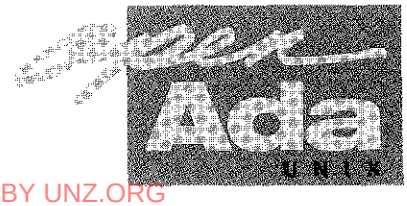
At any price, OpenAda UNIX is an investment in your future. Meridian's renowned customer support is always available. OpenAda is also provided for both DOS and Macintosh, starting at $149.

Order now to begin opening up your opportunities in the Ada/UNIX market. **Call 1(800)221-2522.**

Meridian Software Systems, Inc., 10 Pasteur St., Irvine, CA 92718

OpenAda is a registered trademark of Meridian Software Systems, Inc. All product names are registered trademarks of their respective manufacturers

*According to the DoD Congressional Appropriations Act of 1991, Section 8092. **Available in 60 days. Inquire about pricing.



Circle 59 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

New Turbo Pascal for Windows Don't Leave DOS Without It!

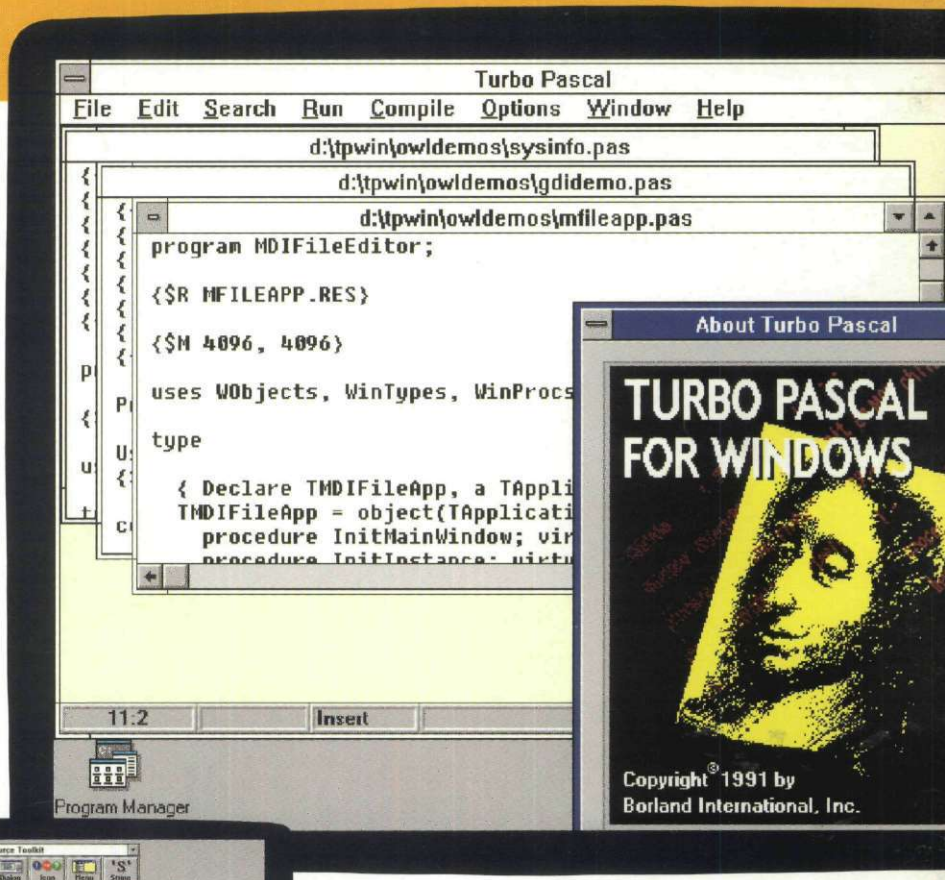
Go with Borland, the leader in OOP and Windows programming, when you're heading for the Windows frontier. With Turbo Pascal® for Windows, your Windows applications will be faster and easier to create.

Turbo Pascal for Windows includes Borland's new ObjectWindows™ application framework FREE. So now you can develop Windows applications fast because they automatically inherit code for windows, menus, dialogs, controls, and more.

Create Windows Applications for Less

Turbo Pascal for Windows gives you more *and* costs you less than other Windows development systems. It's designed exclusively for Windows programming, and everything you need is included in the one low price. You don't need to buy the Microsoft® Windows Software Development Kit (SDK).

Turbo Pascal for Windows is the easiest way to make your next program a Windows program.

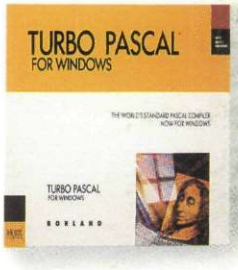


▲ **Create Windows Applications in Windows.** *The Windows Integrated Development Environment (IDE) lets you create, edit, compile and run your programs, all from within Windows.*

◀ **Create Windows Resources Visually.** *Using the Resource Toolkit, you can visually create your Windows user interfaces without programming.*

See Your Dealer Today or Call 1-800-331-0877 Now.

Current owners of Turbo Pascal, call Borland for a special offer!*



BORLAND

The Leader in Object-Oriented Programming for Windows and DOS

CODE: MF28

*Offer good in U.S. and Canada only. Copyright © 1991 Borland BI 1396

Circle **60** on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED